A SIGNAL PROCESSING ALGORITHM BASED ON MULTIPLE MICROPROCESSORS FOR AN UNDERWATER ACOUSTIC IMAGING SYSTEM

Guy Ronald Arthur Vermander

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Signal Processing Algorithm
Based on Multiple Microprocessors
for an Underwater Acoustic Imaging System

by

Guy Ronald Arthur Vermander

December 1980

Thesis Advisor: G. L. Sackman
Co-Advisor: U. R. Kodres

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A Signal Processing Algorithm Based on Multiple Microprocessors for an Underwater Acoustic Imaging System	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, December 1980 / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Guy Ronald Arthur Vermander	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: December 1980 / 13. NUMBER OF PAGES: 85
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Naval Postgraduate School, Monterey, California 93940	15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Signal Processing, Microprocessors, Underwater Acoustics, Acoustic Imaging

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An algorithm has been designed to provide near real-time location and classification information to an operator of an underwater acoustic imaging system. The overall system to detect objects buried up to five meters in unconsolidated marine sediments consists of a specialized projector co-located with a five meter receiving line array. This system is described in basic terms together with the principles of the signal processing that extracts information from the scattered acoustic signals. Using simulated objects, the

angular resolution and imaging performance of a number of system designs were evaluated. To obtain near real-time execution, the algorithm was optimized and partitioned using parallel, pipeline, and double buffering techniques for independent but synchronized operation on multiple microprocessors. The feasibility of the design approach was experimentally demonstrated using four single board computers in a microcomputer development system.

Approved for public release; distribution unlimited

A Signal Processing Algorithm Based on Multiple Microprocessors for an Underwater Acoustic Imaging System

by

Guy Ronald Arthur Vermander
Captain, Canadian Forces
B.S., Royal Military College of Canada, 1966

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

and

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December, 1980

ABSTRACT

An algorithm has been designed to provide near real-time location and classification information to an operator of an underwater acoustic imaging system. The overall system to detect objects buried up to five meters in unconsolidated marine sediments consists of a specialized projector co-located with a five meter receiving line array. This system is described in basic terms together with the principles of the signal processing that extracts information from the scattered acoustic signals. Using simulated objects, the angular resolution and imaging performance of a number of system designs were evaluated. To obtain near real-time execution, the algorithm was optimized and partitioned using parallel, pipeline, and double buffering techniques for independent but synchronized operation on multiple microprocessors. The feasibility of the design approach was experimentally demonstrated using four single board computers in a microcomputer development system.

TABLE OF CONTENTS

I. INTRODUCTION ---------- 7

II. THE PROPOSED ACOUSTIC IMAGING SYSTEM ---------- 9

A. INTRODUCTION ---------- 9

B. TRANSMISSION ---------- 9

C. RECEPTION ---------- 9

1. Physical Configuration ---------- 9

2. Wavefront Curvature ---------- 10

3. The "Trace" of a Point Scatterer ---------- 12

D. SIGNAL PROCESSING ---------- 15

1. The Basic Concept ---------- 15

2. Data Structures ---------- 16

III. THE DEVELOPMENTAL ALGORITHM ---------- 19

A. INTRODUCTION ---------- 19

B. DESCRIPTION OF THE ALGORITHM ---------- 19

1. Parameters ---------- 19

2. Calculation of Traces ---------- 21

3. Simulation of Objects ---------- 23

4. Forming the Amplitude Array ---------- 25

5. Graphics Display ---------- 25

C. RESULTS ---------- 26

1. Range Resolution ---------- 26

2. Bearing Resolution ---------- 27

3. Simulated Images ---------- 30

D. CONCLUSIONS ---------- 34

IV. THE NEAR REAL-TIME ALGORITHM ---------- 36

A. INTRODUCTION ---------- 36

B. THE TIME PROBLEM ---------- 36

C. ALGORITHM OPTIMIZATION ---------- 37

D. ALGORITHM PARTITIONING ---------- 38

1. Parallel Processing ---------- 38

2. Pipeline Processing ---------- 39

3. Semaphores ---------- 39

E. EXPERIMENTAL DEMONSTRATION ---------- 41

1. Equipment ---------- 41

2. The Processes ---------- 41

3. Results ---------- 47

V. CONCLUSIONS ---------- 52

VI. RECOMMENDATIONS ---------- 53

APPENDIX A THE DEVELOPMENTAL ALGORITHM ---------- 55

APPENDIX B THE ALGORITHM IN PL/I-80 ---------- 68

APPENDIX C THE OPERATIONAL ALGORITHM ---------- 78

LIST OF REFERENCES ---------- 84

INITIAL DISTRIBUTION LIST ---------- 85

I. INTRODUCTION

The specific situation which led to this research is the desire to detect, locate, and classify objects buried in unconsolidated marine sediments up to a depth of 5 meters. To accomplish this an acoustic imaging system is needed which can be mounted on an underwater vehicle maneuvering close to the ocean floor. The problem is to penetrate the soft sediment acoustically, detect the scattered signals, and derive from them information as to the nature of the scattering objects. With sufficient range and angular resolution it would be possible to adequately localize and classify an object. Furthermore, it is necessary to obtain this information as rapidly as possible: ideally a display would provide a "real-time" representation of buried objects to an operator as a section of sediment is being searched acoustically.

Although a number of techniques of acoustic imaging exist [1], a system similar to a high resolution sonar has been proposed which would have high resolution using very short baseband pulses and which would operate with dynamic focussing deep in the near field of the receiving aperture. This system would involve a specialized parametric projector co-located with a receiving horizontal line array of hydrophones. It would be mounted on an underwater vehicle operating at about 5 meters above the ocean floor, and would provide acoustic penetration of the marine sediments to depths of about 5 meters. The objective of this thesis was to design the signal processing algorithm for such a system.

In the course of the development of the algorithm the effect of a number of system parameters on angular resolution and imaging performance was evaluated for a number of possible implementations of the acoustic system. This led to major decisions as to the form of the algorithm and the possible acoustic imaging system.

Since an acoustic imaging system mounted on a maneuvering underwater vehicle would be required to provide an operator with a near real-time display of buried objects, the algorithm was redesigned for increased execution efficiency and implementation on multiple microprocessors. Optimization of the algorithm by removing time consuming operations, together with incorporation of parallel and pipeline processes and double buffering concepts were the approaches taken. Experiments were conducted using a number of single board computers to test and demonstrate the feasibility of near real-time execution of the algorithm.

Section II of the thesis describes the acoustic imaging system and the basis for the signal processing algorithm. The details of the algorithm and the angular resolution and imaging performance evaluations are explained in Section III. The design of the algorithm for near real-time execution on multiple microprocessors and the experimental demonstration of feasibility are contained in Section IV. The resulting conclusions are then given in Section V, followed in Section VI by recommendations for future work that could produce a special purpose processor for the acoustic imaging system.

II. THE PROPOSED ACOUSTIC IMAGING SYSTEM

A. INTRODUCTION

An acoustical imaging system mounted on an underwater vehicle will consist essentially of three major parts: a sophisticated projector, a receiving array, and a signal processor and display system. The thrust of this thesis has been to develop the signal processing algorithm. To understand the underlying concepts, the potential form of the projector and receiving array is briefly described, followed by the proposed algorithm.

B. TRANSMISSION

Acoustic waves can penetrate sediment to useful depths with acceptable absorption loss if the frequency is low enough; less than about 20 kHz. Very short acoustic pulses (50 to 100 microseconds) are required for adequate range and angular resolution. For the concept that has been developed in this thesis it has become apparent that it would be necessary to have effectively a point source for the projector. Baseband pulse generation using a parametric source [2] appears to be a promising technique for this application. Such a projector would form a pencil beam in the collimated region of its near field that could be scanned to produce a tomographic image of a region in the sediment.

C. RECEPTION

1. Physical Configuration

The scattered acoustical energy would be received by a horizontal line array (mounted on the underwater vehicle) of a large number of

equally spaced hydrophones. The projector would be located at the center of the line array. The length of the array would be a compromise between the need for a large aperture (on the order of 10 meters for angular resolution) and the physical reality of deploying and maneuvering an underwater vehicle with the array attached (5 meters probably the maximum).

Moving over the area of interest the underwater vehicle would maintain an altitude of approximately 5 meters above the sediment. As the desired depth of penetration into the sediment is also 5 meters the total range of interest lies between 5 and 10 meters from the center of the array. A 60 degree sector (between array bearings of -30 and +30 degrees from the normal to the line array) provides horizontal coverage of 10 meters in the sediment at 10 meters range from the center of the line array (Figure 1).

2. Wavefront Curvature

For ranges between 5 and 10 meters, operation of the acoustical imaging system is deep in the near field of the line array. The far field begins at a range of D^2/λ where D is the diameter of the aperture and λ is the wavelength [1]. Using a 50 microsecond pulse and a 5 meter line array this is on the order of 300 meters. The near field operation is evident by the marked curvature of the wavefronts of the acoustic energy as they arrive at the line array after scattering from points within the area of interest. Since the curvature depends on the range to a scattering point an imaging processor requires some form of dynamic focussing.

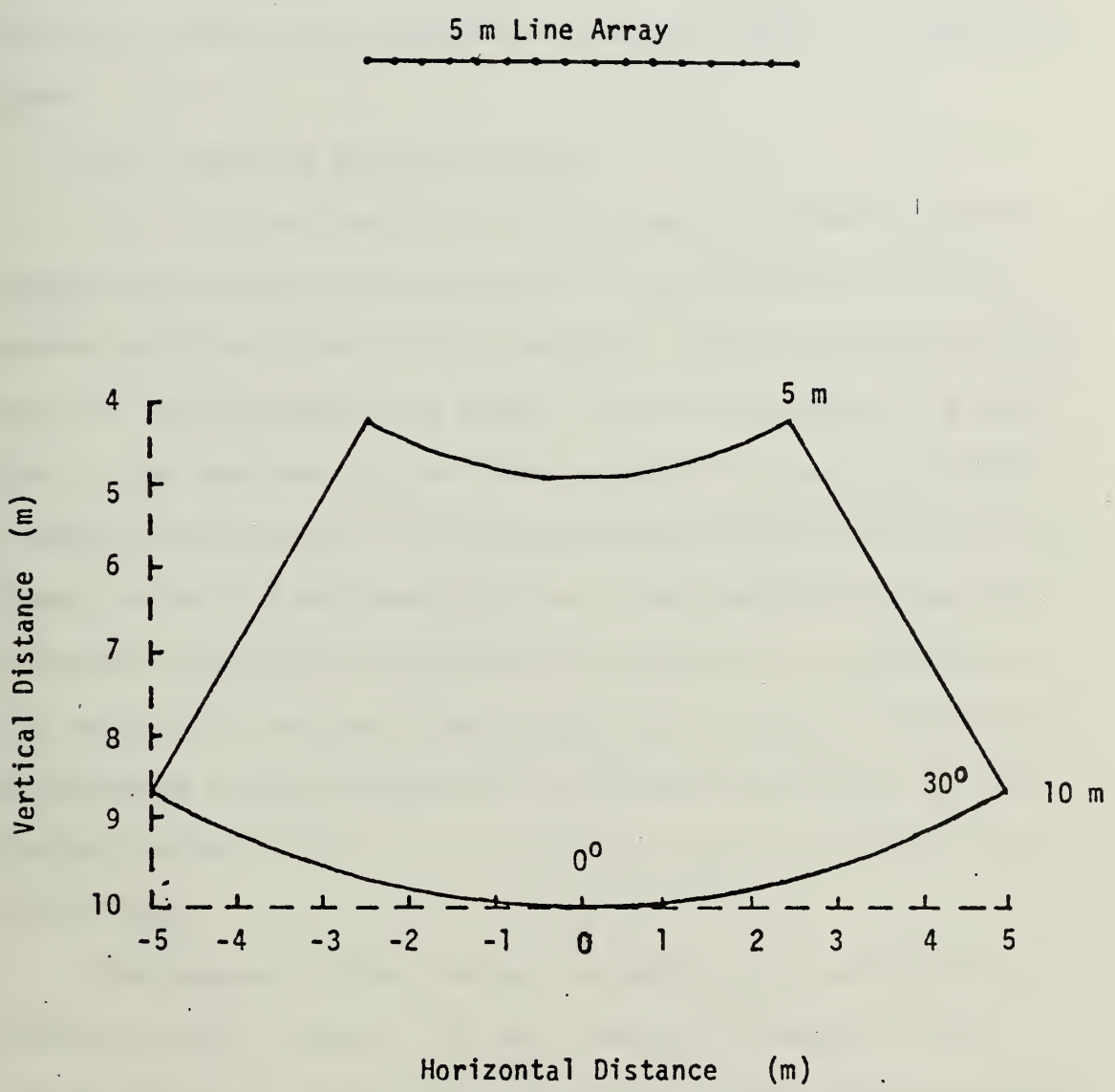


Figure 1. The physical relationship between the five meter line array of hydrophones and the sector of interest.

Figure 2 depicts four such wavefronts (that originated at the indicated bearings and ranges) as a function of time of arrival and position along a 5 meter line array. Using 1500 meters per second as the soundspeed, the time of the earliest arrival of interest is 6.220 milliseconds after pulse transmission, and the latest is 14.304 milliseconds.

3. The "Trace" of a Point Scatterer

The line array consists of a large number of equally spaced discrete hydrophones (each considered as a point) which detect the pressure amplitude of an arriving wavefront. The time sampling of the outputs of 16 hydrophones in a 5 meter line array resulted in a quantization of the wavefront arrival times as shown in Figure 3 (in which an acoustic pulse width of 50 microseconds and sampling every 50 microseconds, beginning 6 milliseconds after pulse transmission, was used). For example, considering the wavefront originating at a scattering point located at a bearing of zero degrees and a range of five meters, the hydrophone sampling produces the following time indices for the wavefront arrivals: 21, 19, 17, 16, 15, 14, 13, 13, 13, 13, 14, 15, 16, 17, 19, 21.

The sequence of time indices representing the arrival of a wavefront, $\underline{n}(r,\theta) = (n_1, n_2 \ldots n_k)$ for K sensors, is unique for the scattering element at a range r and bearing θ: this sequence is termed the "trace" of that particular point (related to the concept of the "Time Delay Trace Function" [3]). The traces for all points at the range and bearing increments in the area of interest can be pre-calculated and stored for later retrieval.

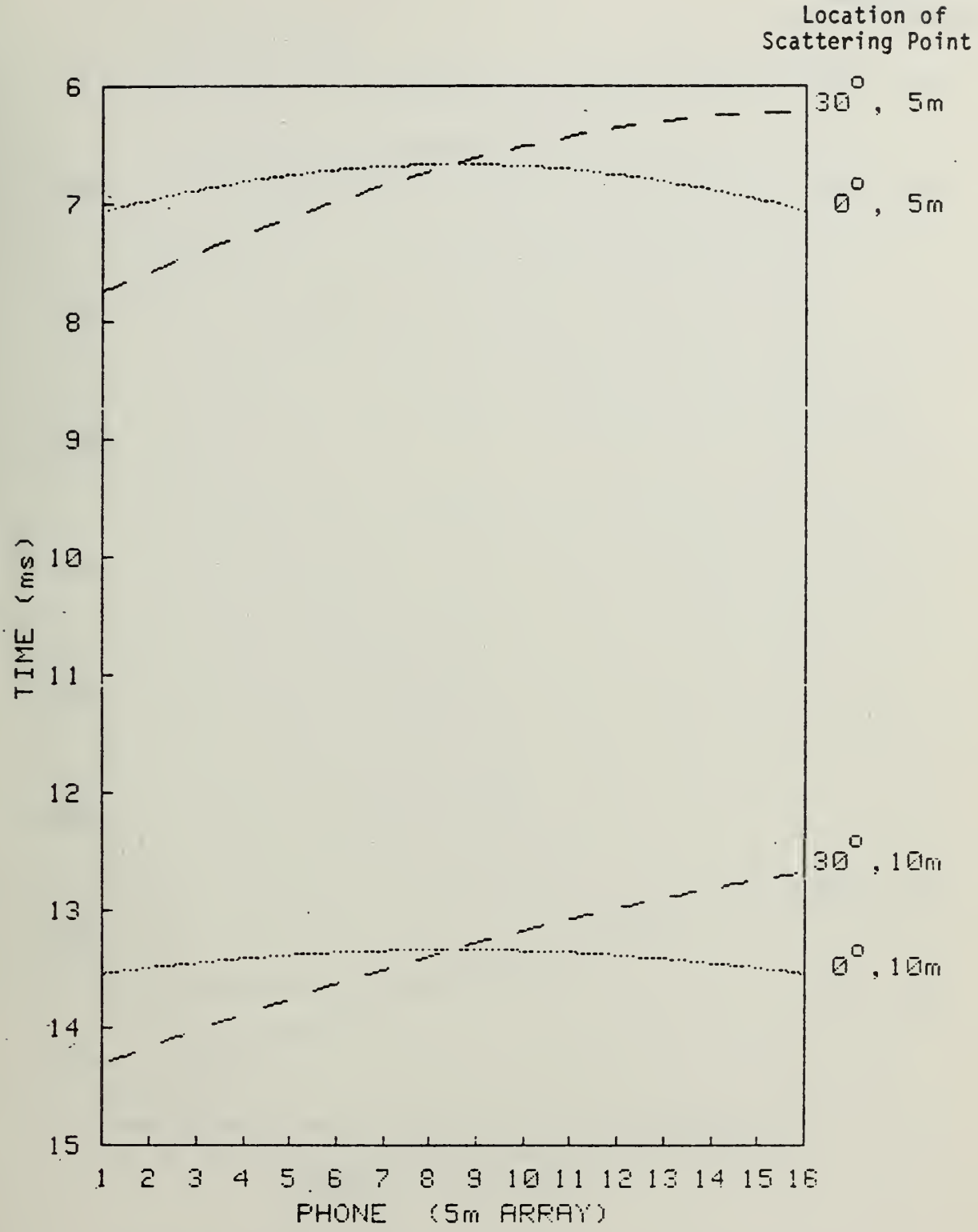


Figure 2. The arrival times of four wavefronts as a function of position along a five meter array.

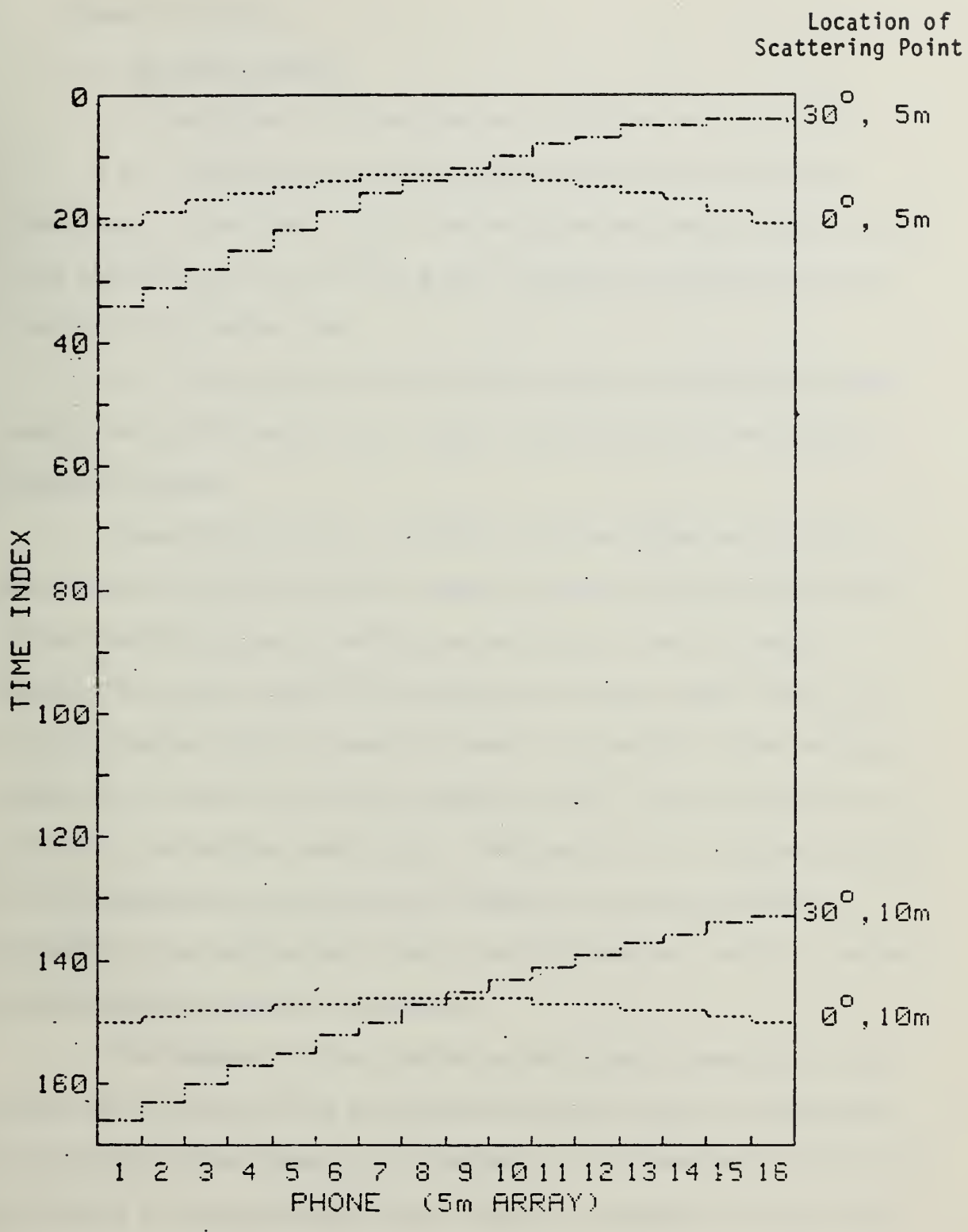


Figure 3. Quantization of wavefront arrival times.

D. SIGNAL PROCESSING

1. The Basic Concept

In developing the algorithm two situations were examined:

a. Insonification of the entire sector by a short pulse (broad beam in the plane of the line array but very narrow beam in the plane perpendicular to the line array) followed by processing returns from the entire sector, and

b. Transmission of the acoustic pulse in a narrow beam and sweeping across the sector with signal processing occurring at each bearing increment.

Essentially, after a suitable gate time after pulse transmission, each sampled hydrophone output provides a time series record of the scattered acoustic amplitudes arriving at the line array. Storing the results (after A/D conversion) at each sample time, n, of all K hydrophone outputs causes the memory to contain a time representation of the scattered acoustic pressure field, D(n,k), k=1..K, n=1..N where N is the maximum sample time. The presence of a scattering point will be indicated by the presence in memory of signal amplitudes occurring at the appropriate times in each hydrophone record (a pattern representing the wavefront curvature).

The sequence of time indices which uniquely represent a scattering point are available as the pre-calculated trace for that point. An estimator of the presence of a pattern is the summation of the signal amplitudes at the appropriate time indices or addresses in each of the hydrophone records. For example, if the signal amplitude at the

wavefront (from a point at r,θ) was considered unity, then the total summation along $\underline{n}(r,\theta)$ would equal K, the total number of sensors:

$$A(r,\theta) = \sum_{k=1}^{K} D(n_k,k) = K$$

where n_k are elements of $\underline{n}(r,\theta)$.

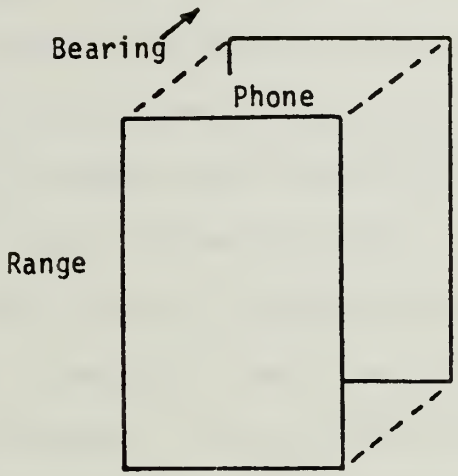
As a result of noise and errors in geometry the sum would be less than K but would nevertheless have a maximum if a scattering point was present.

In summary, what is required is to form in memory a representation of the scattered acoustic field, search through it systematically for wavefront patterns by summation along pre-calculated traces for the ranges and bearings of interest, and recording the resultant total amplitudes as a function of range and bearing. A decision process would then follow to determine which amplitudes represented point scatterers, as opposed to solely noise, and then these could be displayed forming the acoustical image of the insonified region in terms of range and bearing or rectangular coordinates.

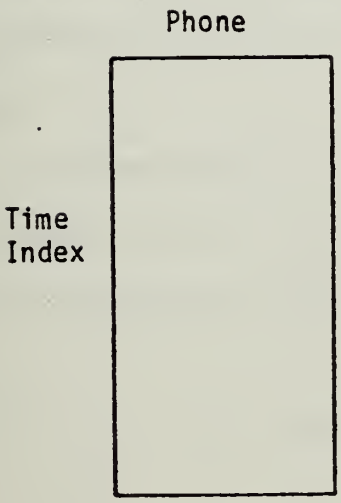
2. Data Structures

To visualize the signal processing operation it is useful to discuss it in terms of the data structures involved: the "Indata" array, the "Trace" matrix, and the "Amplitude" array as shown in Figure 4.

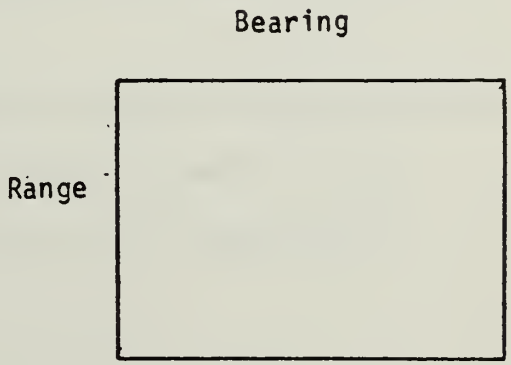
The Indata array contains the time record of the scattered acoustic field. It can be visualized as two dimensional in memory:



Trace Matrix



Indata Array



Amplitude Array

Figure 4. The basic data structures

i.e. D(n,k) where n is the time index, and k is the hydrophone number. Each row would be formed by the simultaneous sampling of all hydrophones at intervals equal to the pulse width of 50 microseconds, beginning after a gate time of 6 milliseconds, and storing the results. This would continue for 167 samples until the last possible arrival time of a signal of interest (time indices 0 to 166).

The Trace matrix, $T(\theta,r,k)$ contains the traces $\underline{n}(r,\theta)$ for points in the area of interest as identified by the intersections of all range and bearing increments. It can be visualized as three dimensional where the first dimension is that of bearing, the second is that of range, and the third is the hydrophone number. For a particular range and bearing, the elements in the third dimension contain the time index sequence $\underline{n}(r,\theta)$ that is the pre-calculated trace: i.e. $n_k = T(\theta,r,k)$.

The dimensions for the Amplitude array are range and bearing. Each element, $A(r,\theta)$, is the total sum of the contents of those elements from the Indata array identified by the trace $\underline{n}(r,\theta)$:

$$\text{i.e.} \quad A(r,\theta) = \sum_{k=1}^{K} D(n_k,k)$$

where n_k are elements of $\underline{n}(r,\theta)$

Scattering points in the acoustically scanned region will be evident by larger amplitude elements in $A(r,\theta)$ and the distribution of these will comprise an image of any objects present.

III. THE DEVELOPMENTAL ALGORITHM

A. INTRODUCTION

The signal processing algorithm was initially developed using a Hewlett-Packard 9845B desktop minicomputer (Figure 5). This machine uses HP enhanced BASIC [4] and was equipped with 192K of storage, two peripheral flexible disk drives (HP9885M and HP9885S), a Graphics ROM (98437B), and a CRT Graphics Memory (98470B). This proved to be a most convenient tool for the development of the algorithm, evaluation of the influence of the major parameters involved, and for the graphical display of simulated objects, all of which led to major conclusions and decisions as to the form of the algorithm and the possible physical acoustic imaging system.

B. DESCRIPTION OF THE ALGORITHM

1. Parameters

The parameters involved in the algorithm (Appendix A) were those which describe the physical implementation of an underwater acoustic imaging system and the area or sector of interest within the sediment that is to be insonified. These parameters were varied to evaluate their influence on the performance of the acoustical imaging system.

The lengths of the horizontal line array used in the developmental algorithm were 10 meters and 5 meters; indicative of the desire for the largest aperture possible for angular resolution, and the most

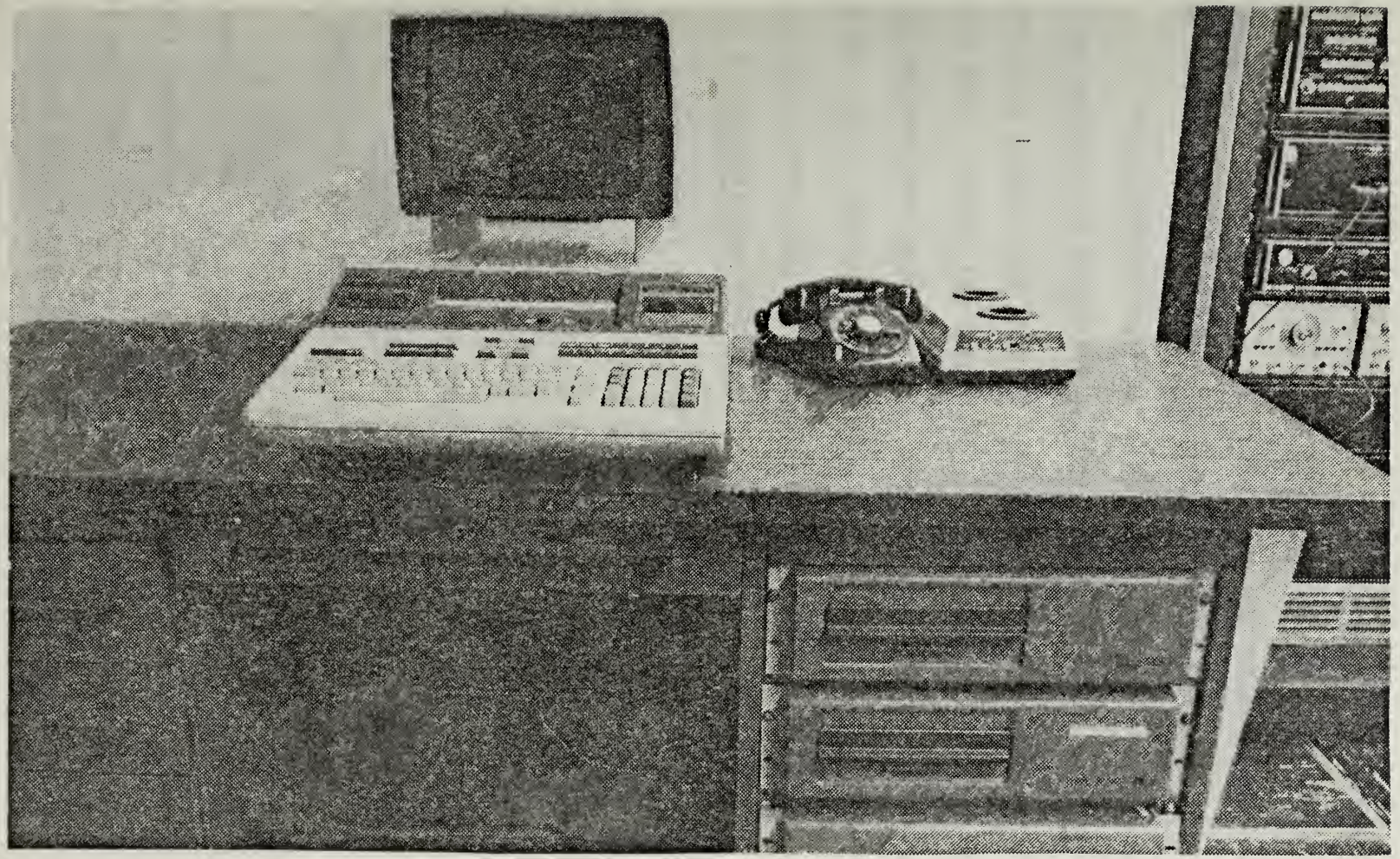


Figure 5. The Hewlett-Packard 9845B

probable realistic maximum for a physical structure to be attached to a maneuvering underwater vehicle. Equally spaced hydrophones along the line array numbering 16 and 32 were used, as were pulse widths of 100 and 50 microseconds. The sound speed was taken as 1500 meters per second throughout the transmission path of the acoustic pulse.

The sector of interest within the sediment was defined as lying between the ranges of 5 and 10 meters from the center of the line array and between bearings of -30 and +30 degrees from the normal to the line array. Range increments of 0.1 meters, and bearing increments of two degrees were chosen. This corresponded to 51 range indices from 0 to 50, and 31 bearing indices from -15 to 15. Bearing increments of one degree were used in the evaluation of array bearing resolution.

2. Calculation of Traces

The trace, $\underline{n}(r,\theta)$, for a scattering point at r,θ consists of the time indices representing the arrival of the scattered wavefront at each of the sensors in the line array. With the projector at the center of the line array, the distance from the projector to each sensor was calculated. A simple trigonometric relation was used to calculate the travel time for an acoustic pulse from the projector to a point $P(r,\theta)$ and then to the k'th sensor.

The geometry of two cases is illustrated in Figure 6 and Figure 7. The first case, Figure 6, applied when the k'th sensor was past the array midpoint; i.e. $k > K/2$ where K is the number of sensors in the line array. For this situation the trigonometrical relation providing the distance d_k from the k'th sensor to the point $P(r,\theta)$ in terms of the range r, bearing θ, and sensor to projector distance b_k, is as follows:

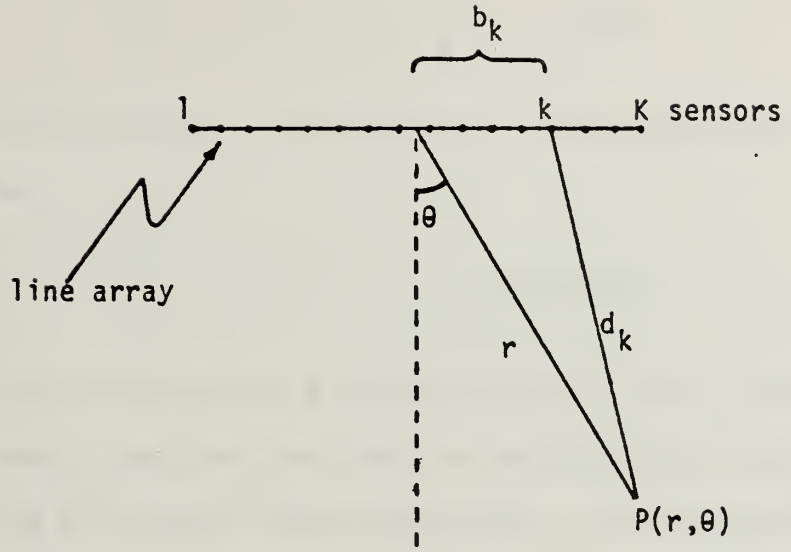


Figure 6. The geometry for k > K/2.

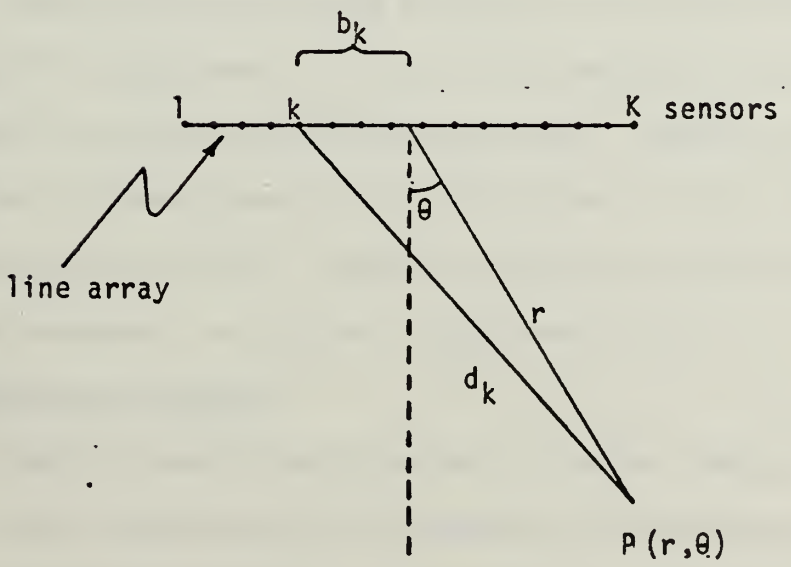


Figure 7. The geometry for k ≤ K/2.

$$d_k^2 = b_k^2 + r^2 - 2\, b_k\, r \cos(90-\theta)$$

Similarly, for the case $k \leq K/2$ (Figure 7), the distance d_k is obtained from

$$d_k^2 = b_k^2 + r^2 - 2\, b_k\, r \cos(90+\theta)$$

The total travel distance for a scattered pulse, $r + d_k$, divided by the sound speed, gives the time that the leading edge of the pulse arrives at the k'th sensor after transmission. The period of 6 milliseconds is subtracted from the travel time, which is equivalent to starting data acquisition after gating out the first 4.5 meters in range from the center of the line array.

The integer portion of the quotient of this time and the sampling interval (equivalent to the pulse width) provides the time index n_k. Computed for each of the K sensors this results in the sequence $\underline{n}(r,\theta) = \{n_k\}$ where k = 1..K. This computation was repeated at all ranges for each of the positive bearings. Symmetry about the normal to the line array permits the calculations of travel times for positive bearings to suffice. Negative bearings use the mirror image of the trace sequence; i.e. $\underline{n}(r,-\theta) = \{n_k\}$ where k = K..1.

3. Simulation of Objects

In order to develop the algorithm and in the absence of real hydrophone inputs, sampling, and I/O operations it was necessary to simulate the Indata array D(n,k). An idealized situation consisting of zero noise and unity signal amplitude at the wavefront of a scattered

pulse was considered. In other words no attempt was made to model a noise environment or the reflectivity of a buried object.

A point scatterer was simulated by using the pre-calculated trace, as stored in the Trace matrix $T(\theta,r,k)$, to identify the time indices n of $D(n,k)$ that represented the wavefront arrival at each sensor. Unity signal amplitude was inserted into these elements. For the narrow acoustic beam case only a single point scatterer was assumed to exist at each range increment along the bearing of the beam. It was more complicated in the case where the entire sector was assumed insonified. In this case the acoustic field would be composed of the interference pattern of numerous wavefronts if the sector contained a distributed object composed of a number of point scatterers.

For the latter case two input situations from the sensors were simulated: a nonlinear input, and a linear input. The nonlinear situation would arise if the outputs from the hydrophones were amplified, passed through a thresholding comparator circuit, and a binary input presented to the computer I/O. The unity amplitude mentioned above simulated this situation. If the sampled outputs from the sensors were subjected to A/D conversion prior to storage in memory, then a more accurate or linear representation of the actual pressure field would be available. This was simulated by the summation of the wavefront amplitudes in the appropriate elements of $D(n,k)$ as each of the points comprising a distributed object was considered.

The angular resolution of the imaging system was evaluated by simulating discrete points in the area of interest. Imaging performance was studied using simulated objects composed of a number of

scattering points. The resulting Indata arrays were then operated upon by the algorithm to form the Amplitude array which represented the acoustic image of the insonified sector.

4. Forming the Amplitude Array

The magnitude of each element in the Amplitude array, $A(r,\theta)$, is an estimator of the presence of a point scatterer at the range r, and the bearing θ. Each element results from the sum of those elements in the Indata array, $D(n,k)$, identified by the trace $\underline{n}(r,\theta)$ stored in the Trace matrix, $T(\theta,r,k)$.

For the case of the entire sector being insonified, the Amplitude array was formed by summing along the traces for all ranges at each bearing in turn for the entire sector. For the narrow beam case, only the traces corresponding to all ranges at the beam bearing were utilized, followed by the acquisition of a new Indata array for the next bearing. Then the traces for all ranges at that bearing were used to guide the summation process, continuing until the narrow beam had scanned the entire sector and the contents of the Amplitude array represented the presence of any point scatterers within the sector.

5. Graphics Display

To evaluate angular resolution and imaging performance of the proposed system the graphics capability of the HP9845B was utilized. It was possible to draw graphs representing the received amplitudes from point scatterers and thereby obtain the system's angular resolution. Both a polar or range-bearing grid and a rectangular grid were devised to provide a means of referencing a displayed image. It was

decided that the rectangular grid would be more practical and meaningful to an operator of an acoustic imaging system.

Due to the choice of 0.1 meter increments in range and two degree increments in bearing, a single point scatterer was represented by graphically "filling-in" an area on the screen equivalent to these dimensions and centered at the range and bearing of the point. This was termed a "target" in the developmental algorithm. A threshold was used to determine when the magnitude of an element in the Amplitude array indicated a target present. An object was simulated by using a number of these discrete points which the graphics display then presented as a number of adjacent targets. This gave the image outline a stepped appearance rather than the smooth lines of a real object.

C. RESULTS

1. Range Resolution

The use of an extremely short pulse of 50 microseconds has inherent range resolution much greater than required or utilized in the development of the algorithm. It was decided that a range resolution of 0.1 meters was adequate to localize and classify large buried objects. Accordingly, it was the range increment chosen for the precalculation of all the traces, and the subsequent processing and display of the acoustic image. An additional consideration was the memory size available to store the Trace matrix.

2. Bearing Resolution

For the case where the entire sector is insonified by one pulse, the effects of a number of parameter changes on the angular resolution of the acoustic imaging system were obtained. The bearing resolution was determined by the ability of the system to distinguish between two discrete points at the same range but at slightly different bearings. The two points were simulated as being in the vicinity of a bearing of zero degrees, and then a bearing of thirty degrees, and at ranges of five meters and ten meters.

The algorithm processed the resulting Indata array in each case and the bearing resolution was determined from the resulting amplitude distribution in bearing at the range of the points. Figure 8 gives an example of the amplitude distribution for two points at a number of bearing separation angles. The amplitude distributions were superimposed as the bearing separation between the points was increased from three degrees to six degrees in one degree increments. The separation in bearing that resulted in 3db and 6db dips between the peaks in the amplitude distribution were recorded and are shown in Table 1.

This was done for both the nonlinear input case and the linear input case, and for sixteen and thirty-two hydrophones in the line array. It was observed that little difference in bearing resolution resulted as could be expected for the idealized zero noise situation that was being simulated.

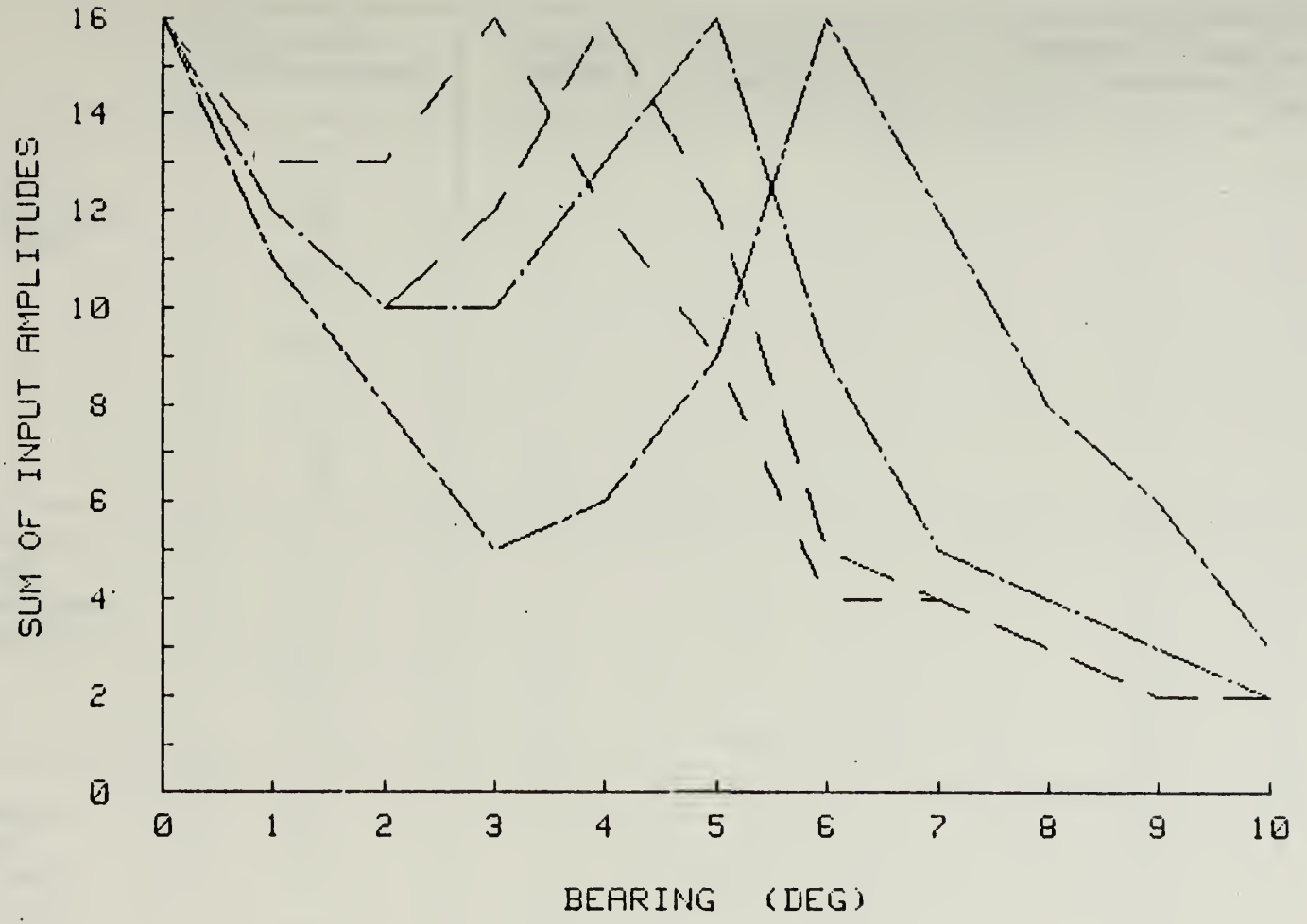


Figure 8. The amplitude distributions for two points at a number of bearing separation angles at 10 meters range from a 5 meter line array.

	Pulse width (microseconds)	Range (meters)	Bearing (degrees)	Resolution (degrees)	
				3db	6db
10 meter array (nonlinear input)	100	5	0	4	5
	100	5	30	4	6
	100	10	0	4	5
	100	10	30	4	6
	50	5	0	3	3
	50	10	0	2	3
5 meter array (nonlinear input)	50	5	0	4	5
	50	5	30	5	5
	50	10	0	4	6
	50	10	30	4	5
5 meter array (linear input)	50	5	0	5	6
	50	5	30	5	7
	50	10	0	5	6
	50	10	30	5	6

TABLE 1. Bearing Resolution as a function of system parameters

3. Simulated Images

Three large objects, each considered as being composed of a number of point scatterers, were simulated in the sector or the field of view of the system. One was small and rounded as if it were a cross section of a long cylindrical object. The other two were long narrow objects; one was in a horizontal position, and the other at an inclined aspect angle. Figure 9 shows the positions and orientations of the simulated objects.

For the case of a pulse insonifying the entire sector and with nonlinear inputs from the receiving hydrophones, the maximum amplitude for any element in the Amplitude array was equal to the number of hydrophones. The threshold for the display of the resulting targets was varied to see the spatial distribution (in range and bearing) of the amplitudes of the image elements. Figure 10 shows the smearing of the objects resulting from the lack of angular resolutions; i.e. the skirts of the amplitude distribution in bearing of each scattering point were being detected.

Similarly Figure 11 shows the results of having linear inputs from the hydrophones. The largest amplitude elements were considerably larger than the nonlinear case, and the contours of the amplitude distribution in the image were much steeper. It was observed that the size, shape and aspect angle of the objects had a substantial effect on the resulting amplitude distribution in the image. This would be beneficial for the identification or classification function of an acoustic imaging system particularly if it provided an operator controlled threshold for the display or if the intensity of the display depicted the amplitudes of the image elements.

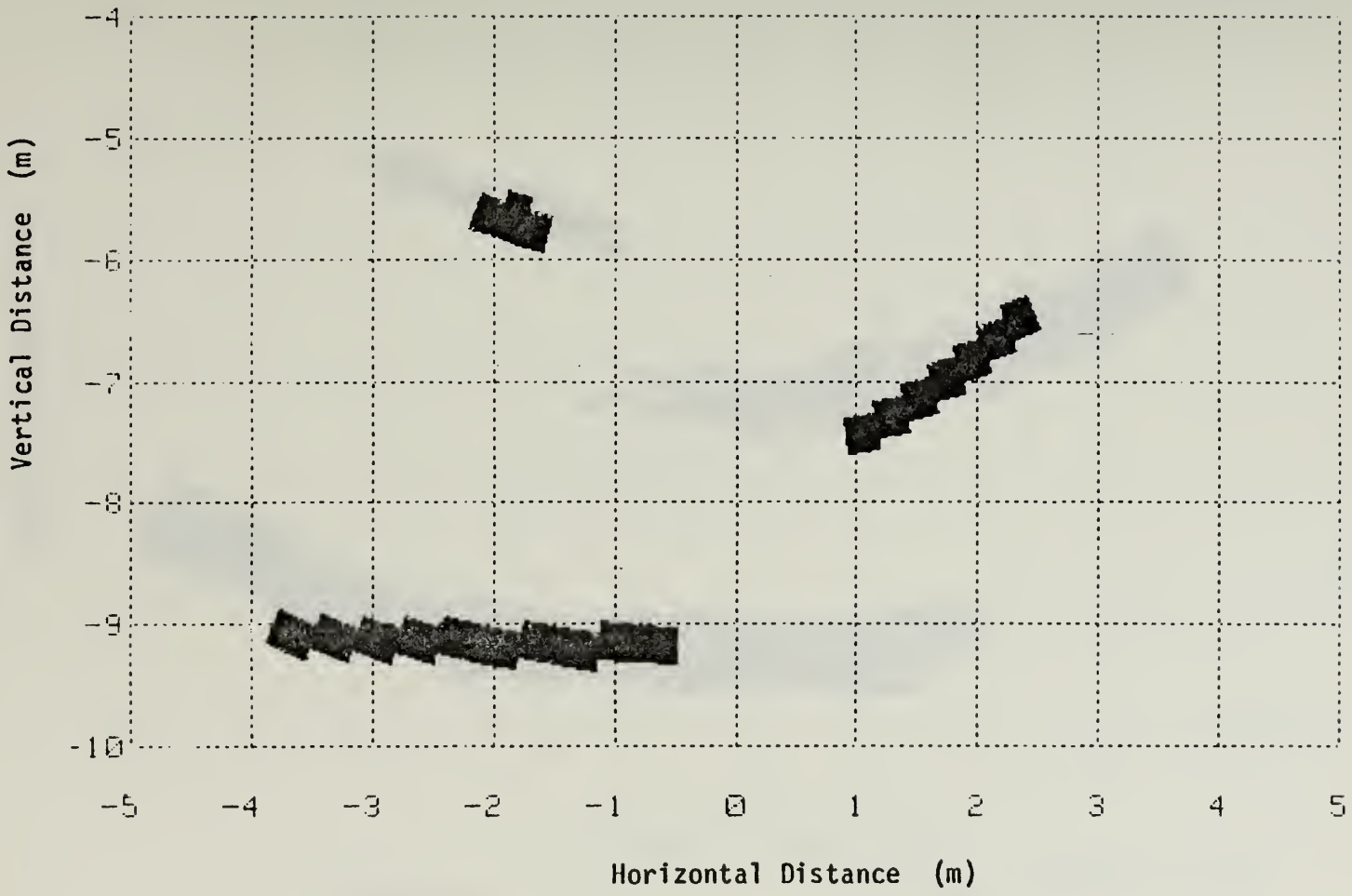


Figure 9. Simulated objects.

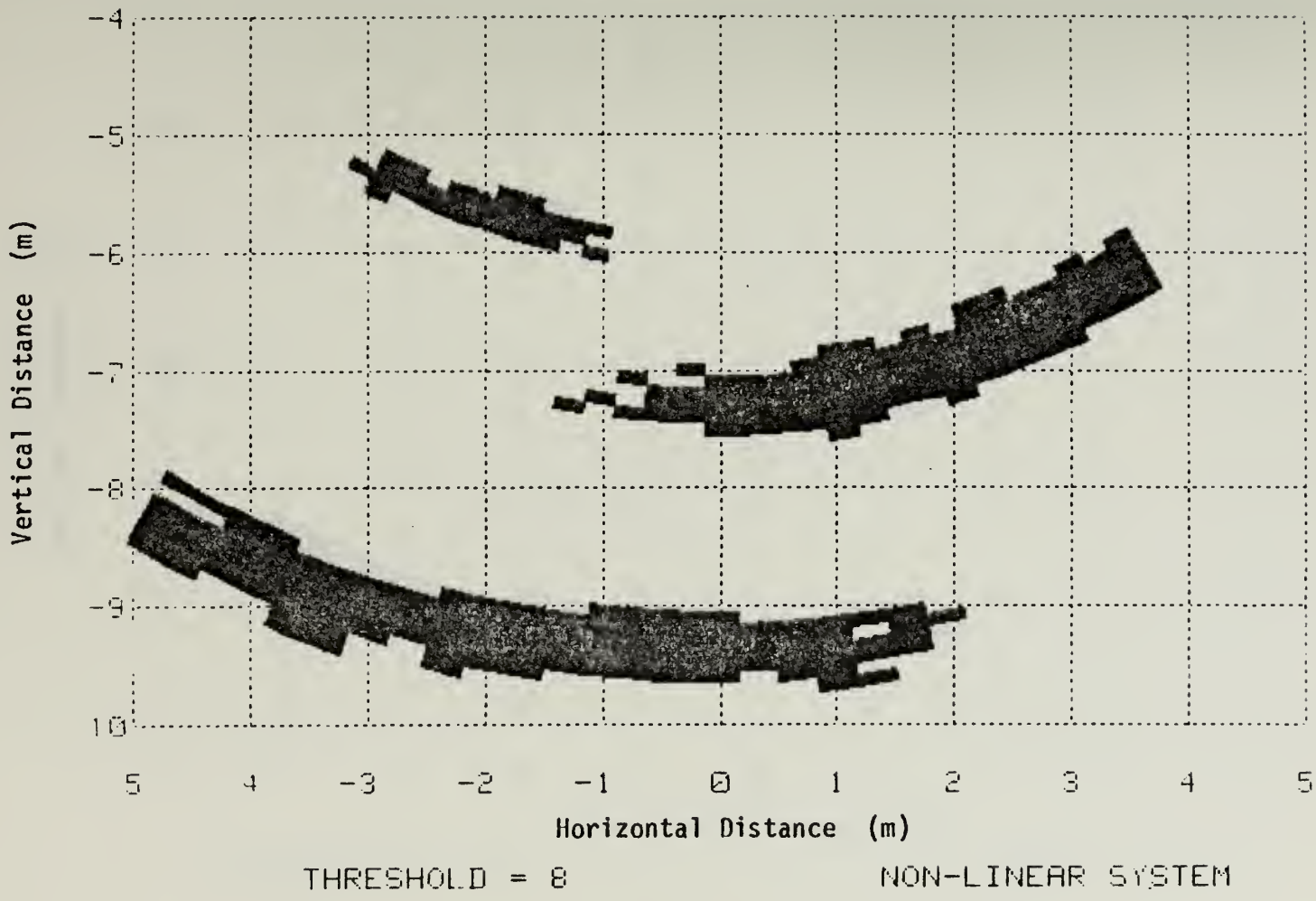


Figure 10. Imaging of the simulated objects by the non-linear system.

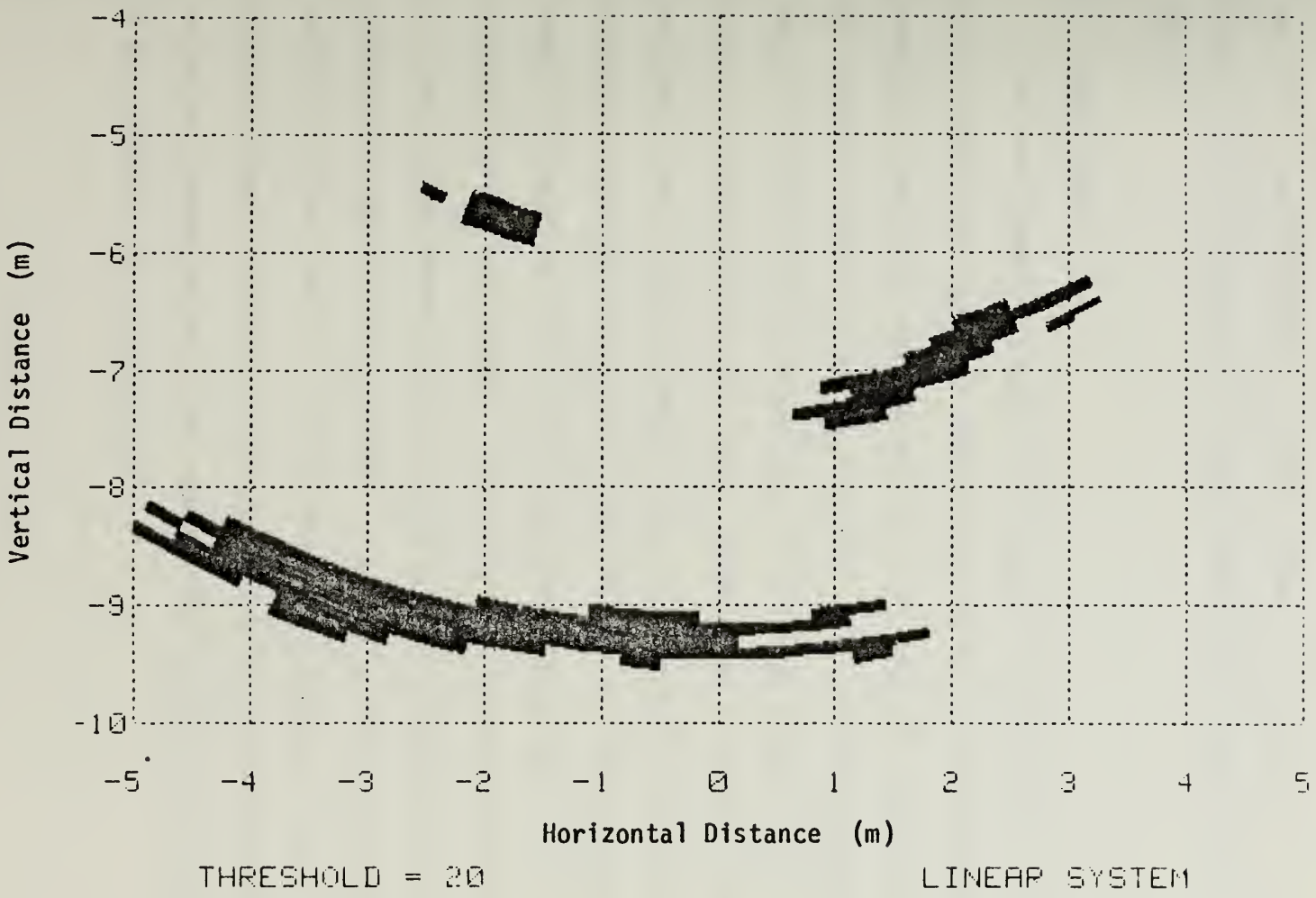


Figure 11. Imaging of the simulated objects by the linear system.

For the case of a narrow two degree beam width system that transmits a pulse at each bearing increment, the resulting image after the sector was completely scanned is shown in Figure 12. It was observed that (as expected) the image exactly duplicated the simulated objects.

D. CONCLUSIONS

The developmental algorithm was used to study some of the effects of the system parameters on bearing resolution, and the imaging performance of possible implementations of an acoustic system that used the proposed signal processing algorithm. It was concluded that the acoustic imaging system should be a narrow beam system that scans the sector by transmitting a very short pulse at each bearing increment, provides linear inputs to memory from the hydrophone line array, and provides an operator controlled threshold for the image display. Although the benefits of the latter two were not evident in the simulated idealized narrow beam images, it is felt that they would be significant in a real system where noise and objects with different scattering or reflective properties were present. This is also the basis for requiring a line array as a large receiving aperture, together with the proposed signal processing algorithm, since objects with specular reflective properties would not necessarily scatter the acoustic energy in the direction of the projector.

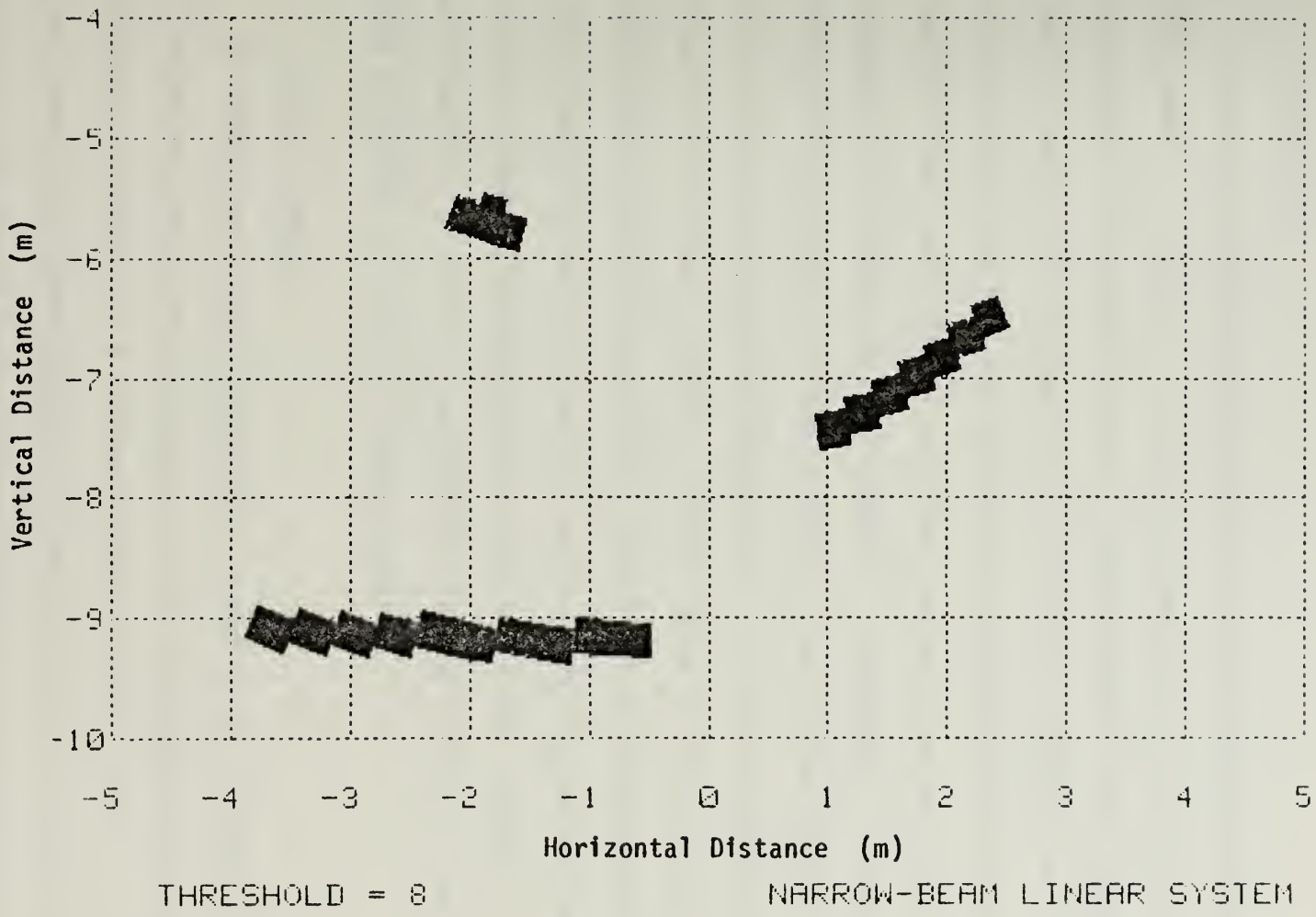


Figure 12. Imaging of the simulated objects by the narrow-beam linear system.

IV. THE NEAR REAL-TIME ALGORITHM

A. INTRODUCTION

An acoustic imaging system mounted on a maneuvering underwater vehicle would be required to provide an operator with a near real-time display of any objects buried in the sediment as the vehicle conducts a search pattern over the ocean floor. Although the signal processing algorithm that was developed was basically simple computationally, it was nevertheless very time consuming and would have been inadequate in an operational environment. Consequently, it was redesigned for increased execution efficiency and implementation on multiple microprocessors while maintaining the basic signal processing approach that has been described. Optimization of the algorithm by removing time consuming operations, together with incorporation of parallel and pipeline processes and double buffering concepts were the approaches taken. Experiments were conducted using a number of single board computers to test and demonstrate the feasibility of the redesigned algorithm for near real-time execution.

B. THE TIME PROBLEM

An underwater vehicle acoustically scanning the sediment and moving at 1 meter per second would probably require a complete sector scan at least once per second as a minimum adequate display rate; i.e. a two degree beam width, approximately 0.34 meters wide at 10 meters range, would scan a 60 degree sector perpendicular to the direction of travel

once every meter of forward travel. At each bearing increment data acquisition would begin 6 milliseconds after pulse transmission and continue for 8.5 milliseconds. Thus the processing of the Indata array can begin 14.5 milliseconds after pulse transmission, and would need to be completed in about 15 milliseconds for each of the bearing increments if a scan were to be completed in about one second. It was obvious that the developmental algorithm would require considerable changes for an efficient microprocessor implementation.

C. ALGORITHM OPTIMIZATION

The algorithm was first examined with the objective of removing time consuming operations such as multiplication. These occurred in the addressing of the multi-dimensional arrays in the algorithm which were stored in row major order. Four additional one dimensional arrays were introduced which provided a table lookup of the row addressing. For example, the addresses for the bearing dimension of the Trace matrix, $T(\theta,r,k)$, were stored in B_array(b), and those for the range dimension in R_array(r). Instead of the addressing for a single element of $T(\theta,r,k)$ requiring three additions and two multiplications, it was addressed using T(B_array(b) + R_array(r) + k); i.e. five addition operations. This was done for each of the multi-dimensional arrays, the Indata array $D(n,k)$, the Trace matrix $T(\theta,r,k)$, and the Amplitude array $A(r,\theta)$, thus reducing the data structures in the algorithm to strictly one dimensional arrays.

With the intention of testing the feasibility of a microprocessor implementation of the algorithm using a number of Intel SBC 80/20 Single

Board Computers (Intel 8080 CPU) [5], the algorithm (Appendix B) was written using the high level language PL/I-80 [6]. Executing the program on one SBC 80/20 it was found that after the pre-calculation of the traces in $T(\theta,r,k)$ the signal processing aspect of the algorithm required 6.45 seconds to complete one sector scan (approximately 210 milliseconds at each bearing, and not including the time required for pulse propagation and data acquisition).

D. ALGORITHM PARTITIONING

The structure of the signal processing algorithm was examined in order to identify functions that could be accomplished by parallel and pipelined processes executing on a number of microprocessors for increased time efficiency. The concept of double buffering was also incorporated to reduce execution time.

1. Parallel Processing

The processing carried out at each of the range increments on the Indata array, for the bearing of the transmitted pulse, is independent of any of the other ranges. This is the summation that results in an element of the Amplitude array $A(r,\theta)$. It was decided to partition this function into separate processes that were identical except for the scope of the range indices. These were called the Sum_amplitude processes and were executed in parallel on a number of microprocessors. Using three microprocessors for example, process S_1 performed summations along the traces for the range indices 0 to 16, process S_2 used range indices 17 to 33, and process S_3 covered range indices 34 to

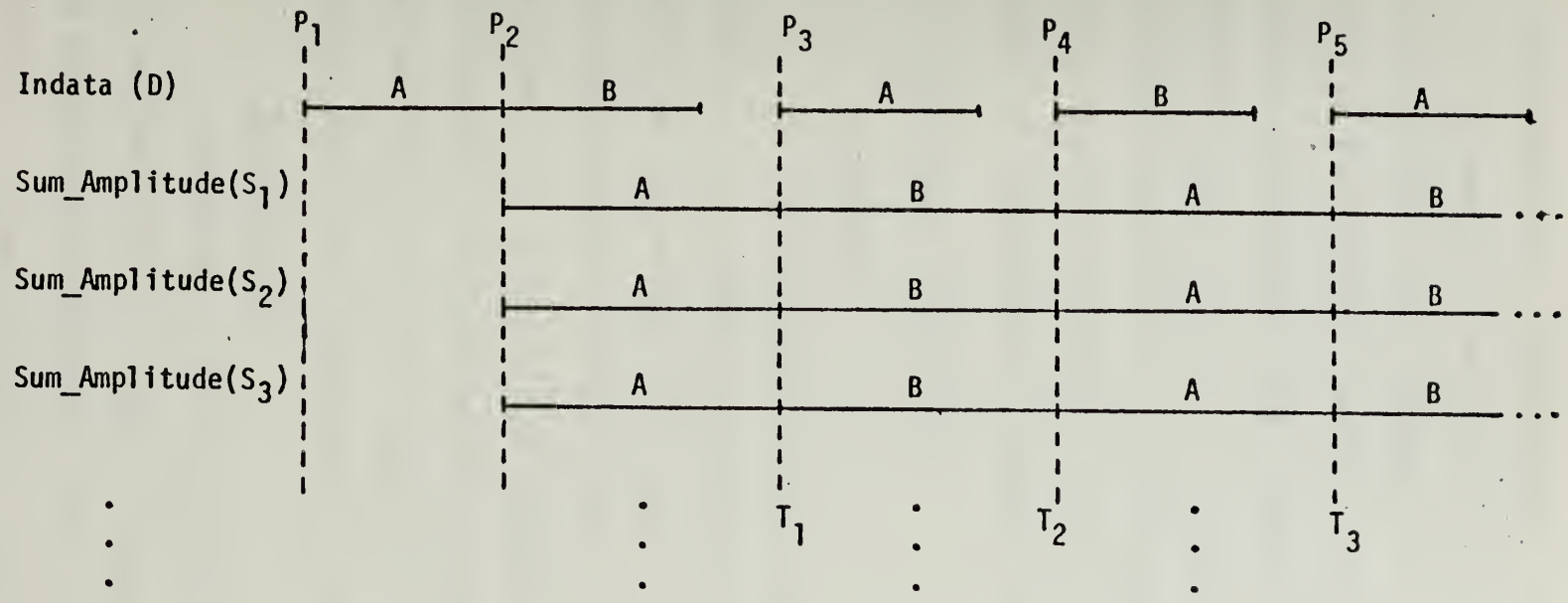
50. In this manner all ranges were processed in one third the time that was required by a single process executing all range increments.

2. Pipeline Processing

The function of data acquisition is called the Indata process and was seen to be amenable to pipeline processing with the Sum_amplitude processes. This process could control the timing of pulse transmission, sampling the hydrophone outputs, and storing the results into two buffer arrays. The Indata array of the developmental algorithm became two arrays called the A_indata array and the B_indata array. After the A_indata array was filled with the data from a particular bearing, the Sum_amplitude processes were permitted to begin their operations on the A_indata array. Simultaneously, the Indata process proceeded to fill the B_indata array with the data from the next bearing increment. When the Sum_amplitude processes had finished with the A_indata array, they processed the B_indata array. The Indata process could then utilize the A_indata array for the next bearing and so on in a pipeline fashion. Figure 13 illustrates the relationship between the processes.

3. Semaphores

Communication between the processes was accomplished with binary semaphores. The Indata process set the semaphores A_data and B_data whenever it was utilizing either the A_indata or B_indata array. Similarly, each of the parallel Sum_amplitude processes set a semaphore depending on the data array it was processing. Thus for three parallel Sum_amplitude processes six semaphores were utilized: A_ampl, A_amp2,



Note: P_i are pulse transmission times

T_i are times for completed processing at each of the bearing increments.

A denotes operations on the A_indata buffer array.

B denotes operations on the B_indata buffer array.

Figure 13. The relationship between the processes.

A_amp3, and B_ampl, B_amp2, B_amp3. The Indata process waited for each of the Sum_amplitude semaphores for one of the data buffer arrays to be unlocked before it could proceed to fill the buffer with new sampled data from the hydrophone outputs. Similarly, each of the Sum_amplitude processes waited on the A_data or B_data semaphore before it began processing the data. In this way completely independent but synchronized operation of all the processes was achieved. The operational form of the algorithm in Appendix C illustrates the partitioning of the processes and the use of the semaphores.

E. EXPERIMENTAL DEMONSTRATION

1. Equipment

To test and demonstrate the feasibility of near real-time execution of the redesigned algorithm an Intellec 800 Microcomputer Development System (MDS) [7], shown in Figures 14 and 15, with the CP/M monitor control program [8] was used. The MDS is capable of being used as a partial system simulator to check out software executing from RAM in the Intellec MDS. A total of 64 kilobytes of memory was available and four Intel SBC 80/20 Single Board Computers [5] were inserted into the MDS motherboard as shown in Figures 16 and 17.

2. The Processes

The contents of the Trace matrix, the pre-calculated traces, were computed using the program of Appendix B, and stored in a floppy disk data file for subsequent loading into memory. The redesigned "operational" algorithm (Appendix C) was written as if it were to be executed on a single processor, and consisted essentially of a Setup

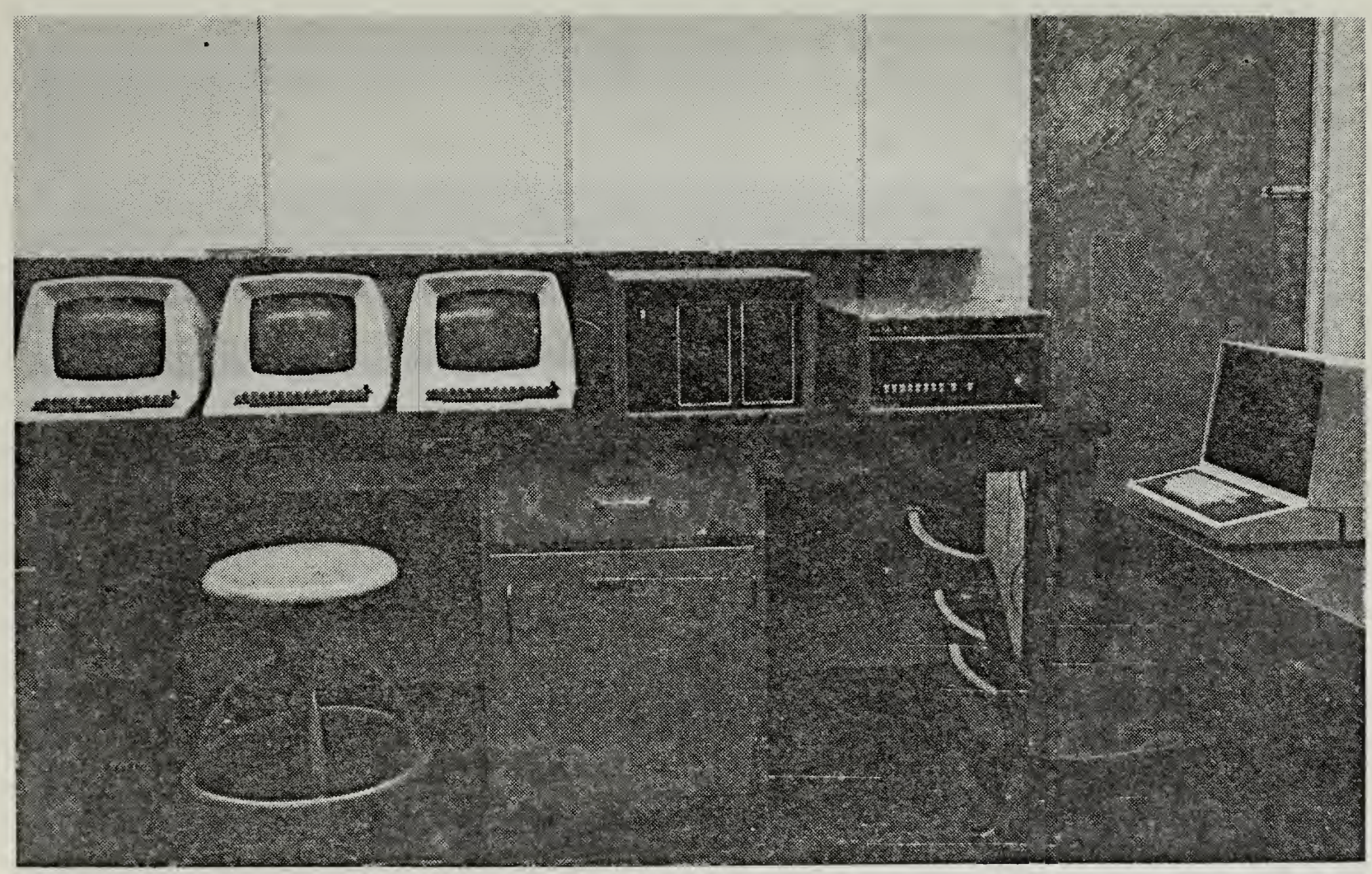


Figure 14. The system used for the experimental demonstration

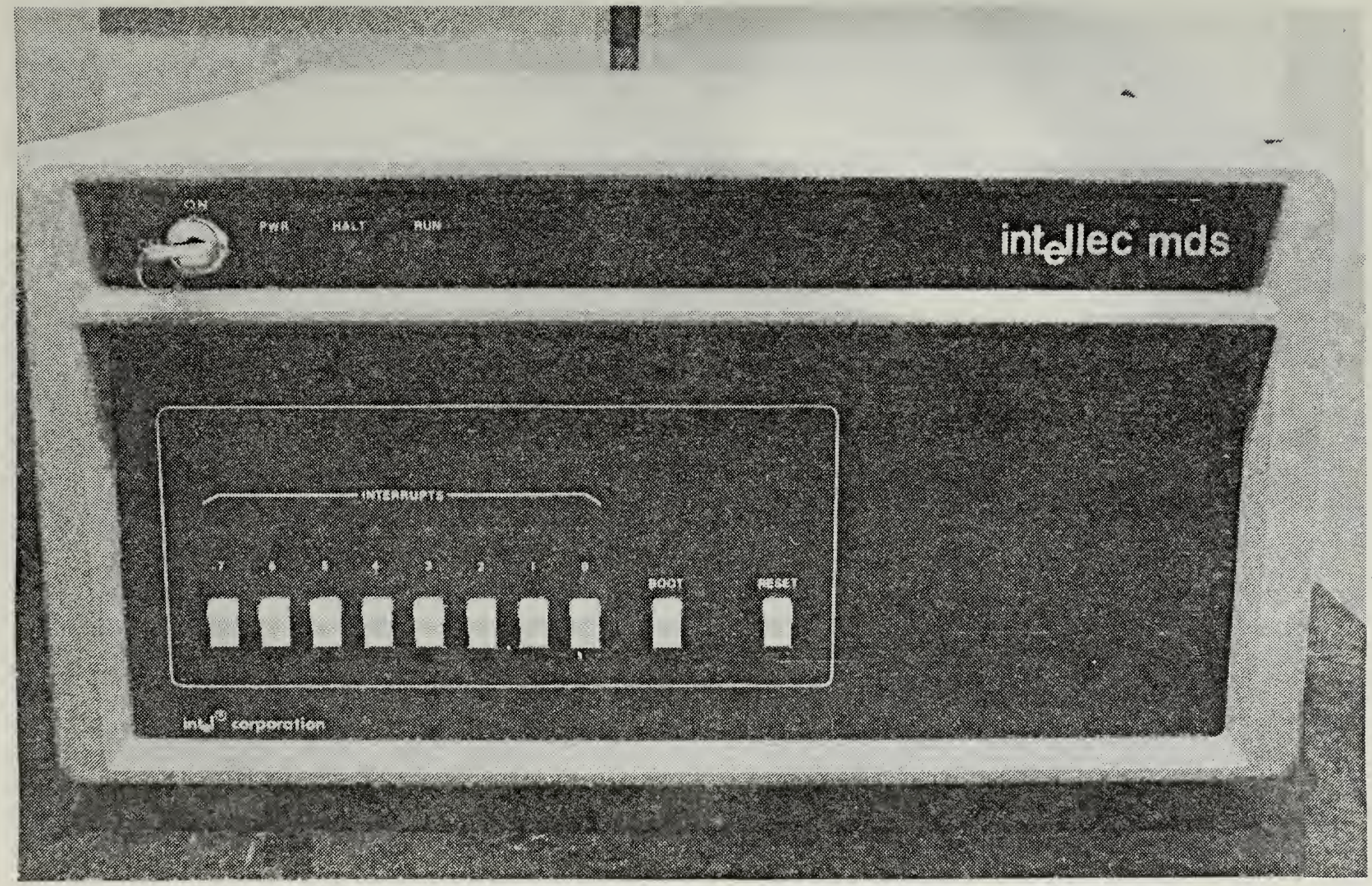


Figure 15. The Intellec 800 Microcomputer Development System (MDS)

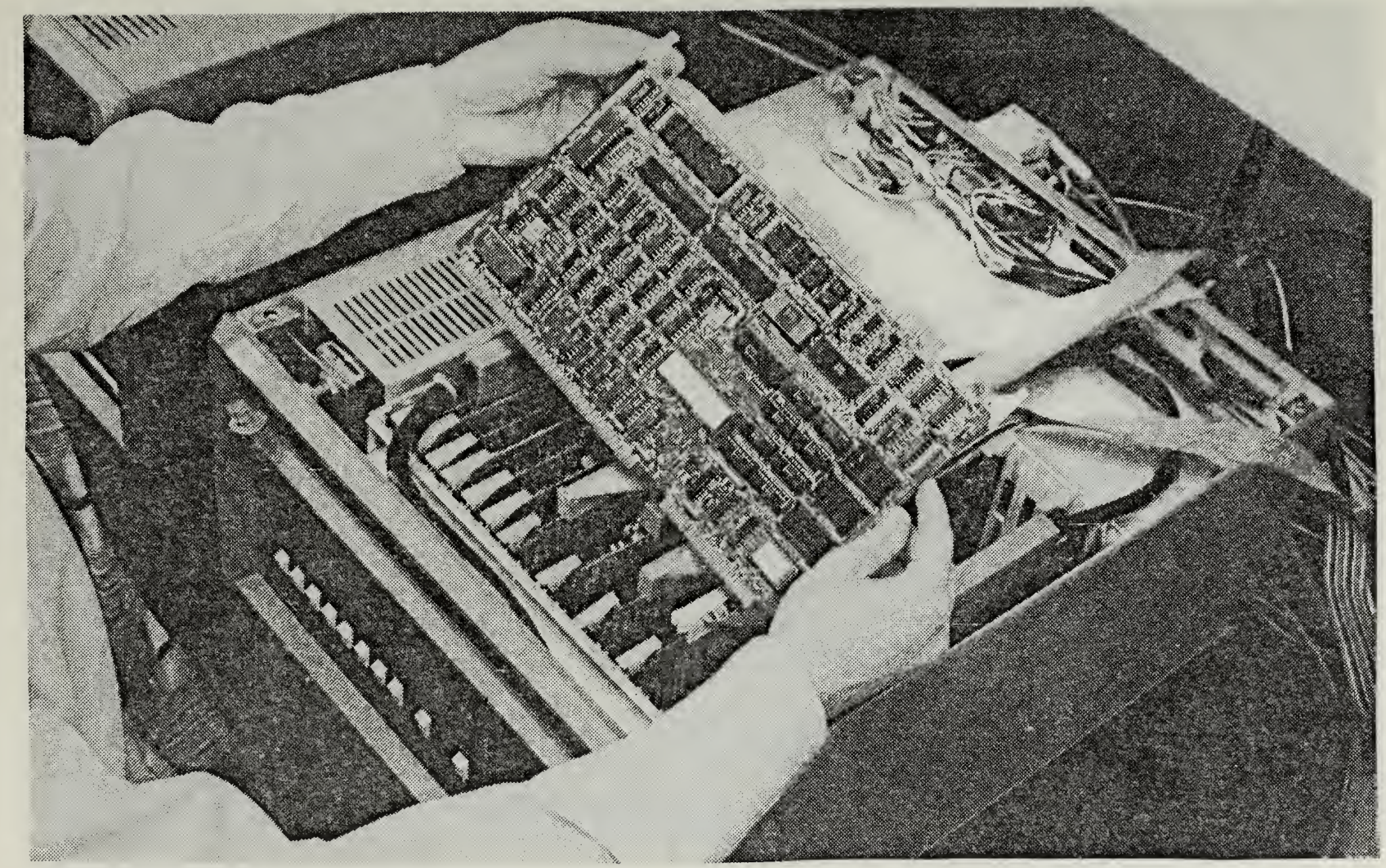


Figure 16. The Intel Single Board Computer SBC 80/20

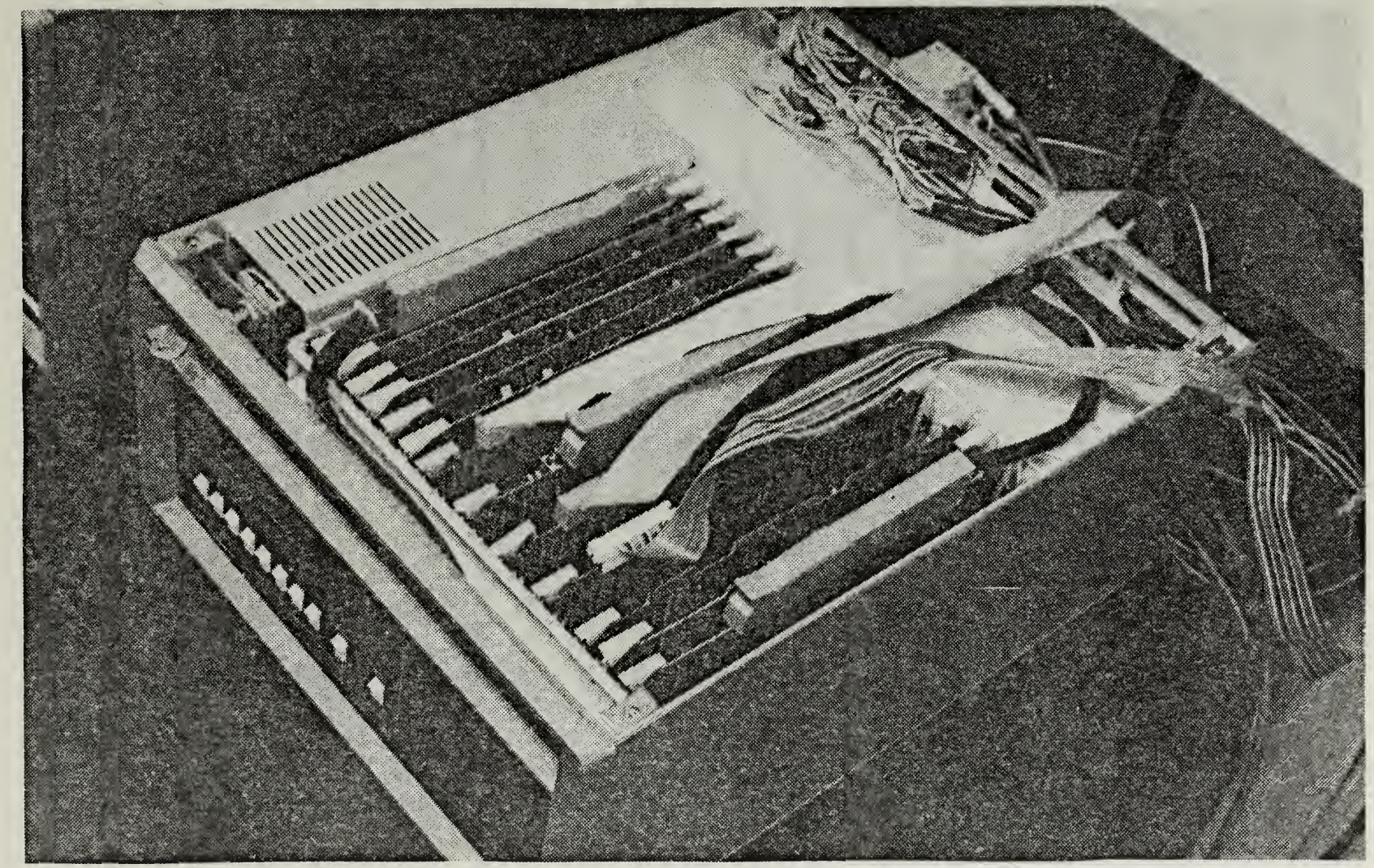


Figure 17. The MDS with 64K of RAM and four SBC 80/20's inserted in motherboard.

procedure that calculated the contents of the address lookup arrays, the Indata procedure, and the Sum_amplitude procedure. This was compiled and linked so the program started at location 3000 Hex. The choice of 3000 H as the initial address for the program was based on the fact that addresses 3000 H to 3EFF H were available as private RAM on each of the SBCs. All the data structures and variables such as the semaphores that were to be global to all the processes were located in common memory above 4000 H that was accessible by all the SBCs. An empty dummy array of the necessary size was declared to ensure the assignment of common memory addresses to the global variables.

Using the MDS and the Dynamic Debugging Tool (DDT) program [9] the program was altered at the machine code level (hex representation) by removing the appropriate call statements and creating two versions: one which executed only the Indata procedure, and the other the Sum_amplitude procedure. In this way the two processes stored in the private memory of separate SBCs could operate independently but in a synchronized manner using the semaphores in common memory. The data that was operated on, as well as all the data structures, were available in common memory.

The Sum_amplitude version was then amended using DDT so that all the local variables in the procedure, which the compiler had assigned to memory locations in the common area, were readdressed so they were within an SBC's private memory. The locations assigned to the dummy array were utilized for this purpose. This then provided a template for three subsequent copies of the Sum_amplitude process each of which was

individually altered to process a segment of the range indices, and then stored on a disk file.

Similarly, the addresses of variables that were local to the Indata procedure were amended so they were within an SBC's private memory. Having loaded the Trace matrix from the floppy disk data file into the appropriate memory locations that had been assigned by the compiler, the entire Indata version (which included the data structures) was saved on a disk file. A subsequent reloading of this file into memory caused the Trace matrix to also be available and so a separate loading of the Trace matrix data file was not necessary. For the experimental demonstration of the feasibility of the multiple microprocessor concept the Indata process controlled the bearing parameter and caused the entire algorithm to scan ten complete sectors and then stop. An operational version would continue indefinitely. As the double buffering design resulted in two bearing increments being processed on each pass through the procedure, it was necessary to ensure that each sector consisted of an even number of bearing increments, i.e. thirty vice thirty-one. No attempt was made to include pulse propagation time or data acquisition time. The procedure that would fulfill these functions was left as a "stub" since it was not required for the demonstration of the synchronization between the processes.

3. Results

The capabilities of the MDS and DDT were used to subsequently load the Indata process into the private memory of one of the SBC's, and the Sum_amplitude processes into the private memory of each of the three remaining SBCs. Figure 18 provides a block diagram of the experimental configuration where process D is the Indata process and processes S_1 to

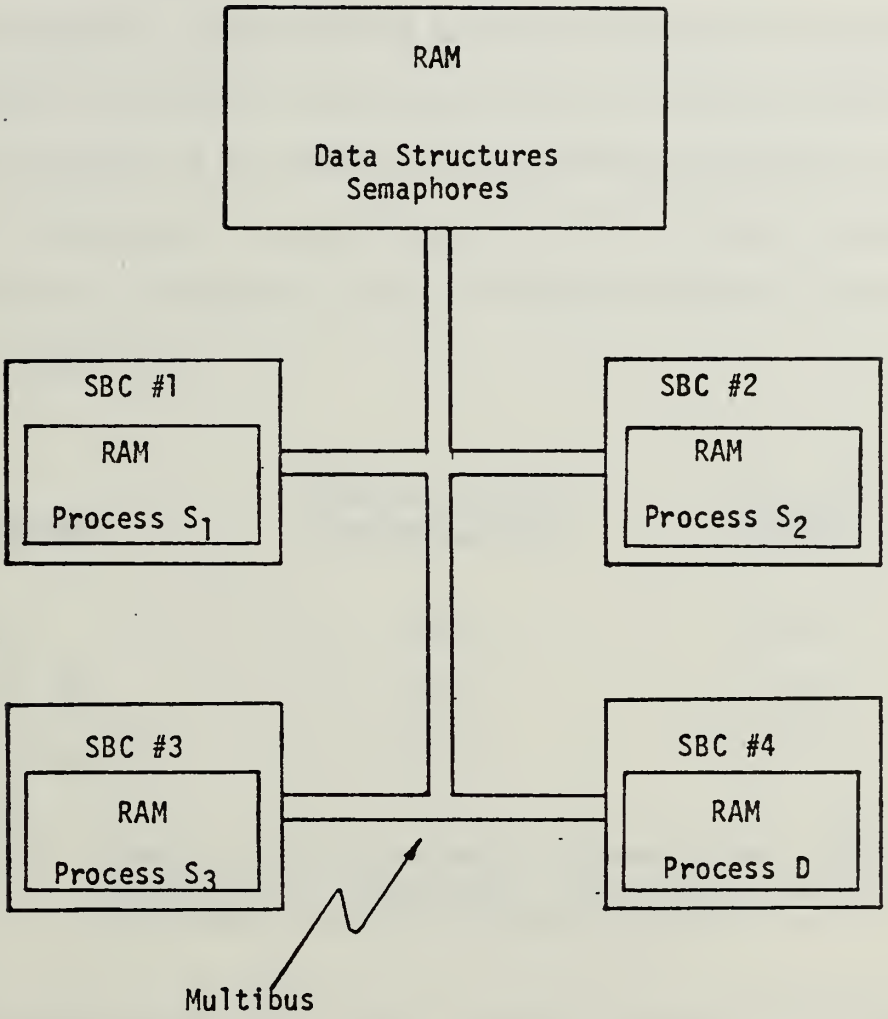


Figure 18. Block diagram of experimental configuration.

S_3 are the Sum_amplitude processes. With the processes executing simultaneously the time required for the amplitude summation function to be performed on ten sector scans was recorded.

Executing the algorithm with a different number of Sum_amplitude processes in parallel, while ensuring that all range indices were processed, permitted an evaluation of the effect of increasing the number of parallel processors. The resulting times for one sector scan and consequently the performance in terms of scans per second are given in Table 2 and Figure 19.

Processes	Time for one scan (seconds ±0.01)	Scans per Second
D, S_1	6.45	0.155
D, S_1, S_2	3.23	0.310
D, S_1, S_2, S_3	2.15	0.465

TABLE 2. Results of the experimental demonstration of the feasibility of implementing the signal processing algorithm on multiple microprocessors.

The correct operation of the algorithm together with the performance times indicated that the design was a feasible approach to obtaining near real-time execution of the proposed signal processing algorithm. The fact that the processes were stored in the SBCs private memory and required access to the bus only when the common data structures and semaphores were referenced resulted in a minimum of bus usage. An examination of the code for the innermost loop of the Sum_amplitude process (the summation of the Indata array elements identified by the

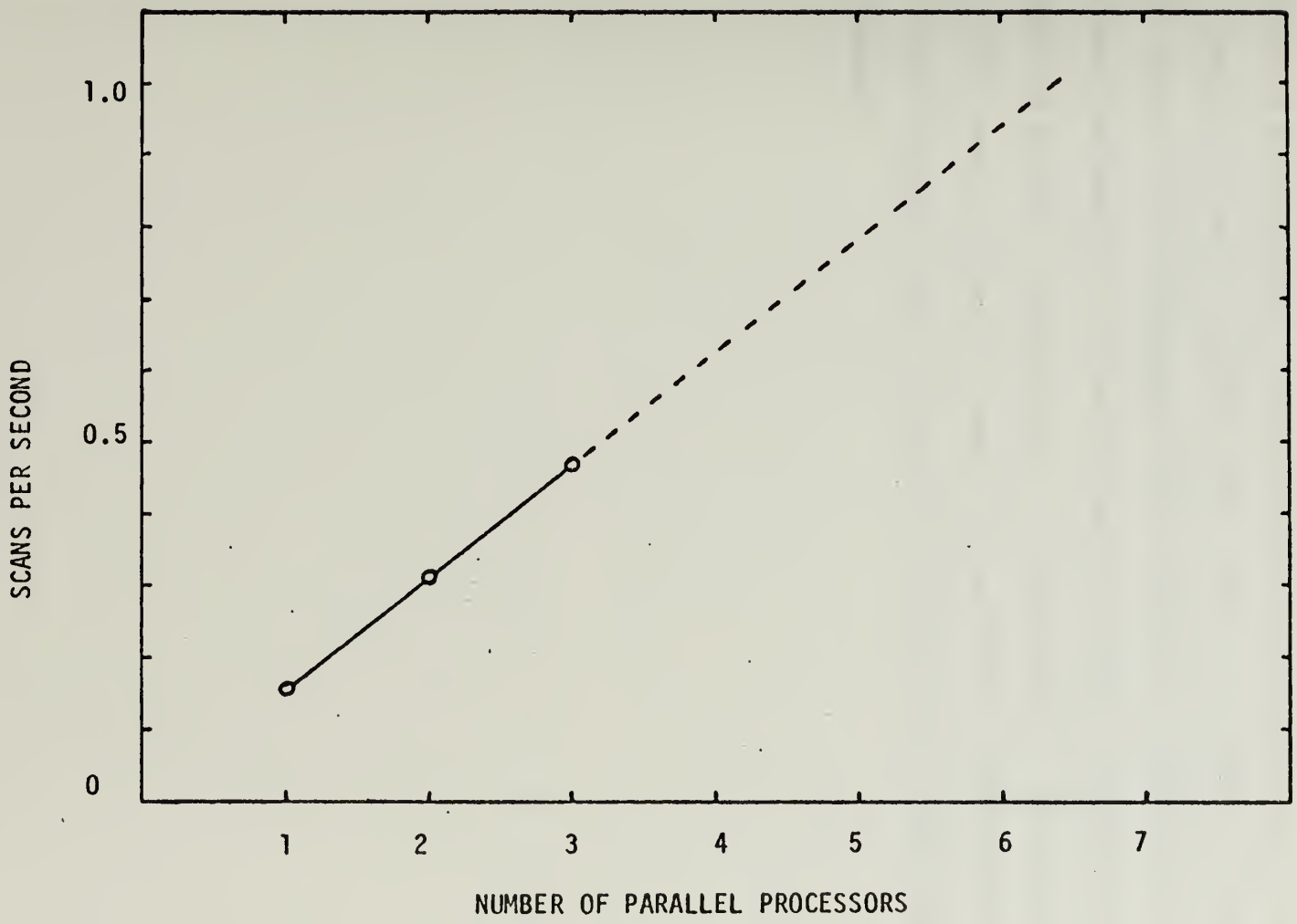


Figure 19. Performance as a function of the number of parallel processors used.

trace) indicated that the bus usage was 7.9% of the loop's execution time. The number of parallel SBCs executing simultaneously could theoretically be increased to twelve before bus contention would adversely influence the overall execution time. Table 2 and Figure 19 depicting the results of the experiment clearly indicate the linear relationship between performance, in terms of scans per second, and the increasing number of parallel processors used. If the number of SBCs operating in parallel were increased to seven, plus one for the Indata process for a total of eight, then a scan rate of one sector per second would be achieved.

V. CONCLUSIONS

For the problem of locating and classifying objects buried in marine sediments an underwater acoustic imaging system has been proposed and the thrust of this thesis has been the design of the signal processing algorithm that would extract the information from scattered acoustic signals for subsequent operator display. Using a developmental form of the algorithm it was determined that the acoustic imaging system should be a narrow beam system for optimum resolution and imaging performance. Since the processing time was excessive, the algorithm was redesigned for implementation on multiple processors for execution efficiency. It was experimentally demonstrated that this design was a feasible approach to obtaining near real-time execution of the algorithm.

The maximum rate of sector scanning that is possible for the proposed system is slightly in excess of two sector scans per second. This was arrived at by considering the pulse propagation time and the data acquisition time, a total of 14.5 milliseconds at each bearing, as the minimum time between pulses. It has been shown that the use of multiple microprocessors can result in an efficient implementation of the proposed signal processing algorithm. The number of parallel processors required to reduce the amplitude summation process to about 15 milliseconds (and thus achieve the maximum rate of two scans per second) would be dependent on the microprocessor chosen.

VI. RECOMMENDATIONS

The signal processing algorithm and its implementation on multiple microprocessors, as discussed in this thesis, would form the core of a special purpose processor for the acoustic imaging system. Two areas would require development and detailed definition before the entire special purpose processor could be designed: data acquisition to provide inputs to the processing algorithm, and output from the algorithm in the form of a display.

It is recommended that the Indata process outlined in the thesis be expanded from the current "stub" status to perform the functions previously mentioned; i.e. control of pulse transmission timing, sampling the hydrophone outputs, and storing the results into the two buffer arrays. It is anticipated that the approaches used for the algorithm design would be readily applicable to these functions. However, the actual details would be highly hardware dependent which would require investigation prior to designing a parallel/pipeline approach to this portion of the specialized processor.

Similarly, the interface between the multiple microprocessor system (performing the signal processing) and a display processor would require definition and would be hardware dependent. It is considered likely that the image information after each sector scan could be stored thus forming a three dimensional representation in memory (the third dimension resulting from the travel of the underwater vehicle

over a section of sediment). This would provide an opportunity for a two dimensional display in either the vertical plane (such as range and bearing) or in a horizontal plane at some range. Furthermore, the possibility of a three dimensional display could be considered. It is evident that there is much promising work to be accomplished in order to produce the special purpose processor for the acoustic imaging system.

```
10                              !           IMAGES
20                              !                 (HP enhanced BASIC)
30                              !
40                              ! FORMS Tracematrix(), Indata(), Amplitude()
50                              !   AND PLOTS TARGETS (5m ARRAY) NARROW-BEAM
60                              !
70    DEG                       ! ALL ANGLES WILL BE IN DEGREES
80    OPTION BASE 0             ! MATRIX INDICES BEGIN AT 0
90    Max_bearing=30
100   Bearing_inc=2
110   Dx=TAN(Bearing_inc/2)     ! USED IN Sector_grid AND Target
120   Min_range=5               ! ALL DISTANCES WILL BE IN METERS
130   Max_range=10
140   Range_inc=.1
150   Arraylength=5
160   Max_phone=15              ! NUMBER OF HYDROPHONES:  DIMENSION OF Base( )
170                             !   MUST BE >= Max_phone
180                             !     (16 PHONES: INDEXED FROM 0 TO 15)
190   Base_inc=Arraylength/Max_phone
200   DIM Base(15)              ! ARRAY CONTAINING DISTANCES FROM CENTER
210                             !   OF LINE ARRAY TO EACH HYDROPHONE
220   Soundspeed=1500           ! METERS/SEC
230   Blank_distance=4.5        ! NEAR SCATTERERS IGNORED/TRANSDUCER PULSE BLANKED
240   Pulse_width=.000050       ! SECONDS  (50 MICROSECONDS)
250   Threshold=8
260   INTEGER Trace_matrix(15,50,15) ! DIM'S ARE: BEARING,RANGE, AND PHONE
270   File$="TRACES"            ! NAME OF MASS STORAGE FILE STORING Trace_matrix
280   INTEGER Objects(50,-15:15)
290                             ! CONTAINS SIMULATED TARGET ELEMENTS
300                             !   -DIMENSIONS ARE RANGE AND BEARING
310   INTEGER Indata(166,15)    ! MATRIX OF THE RECEIVED SIGNALS: DIMENSIONS
320                             !   ARE TIMESLOT AND PHONE
```

```
330   INTEGER Amplitude(50,-15:15)
340                             ! CONTAINS THE SUM OF THE ELEMENTS OF Indata()
350                             !   THAT ARE IDENTIFIED BY Trace_matrix()
360                             !   AS HAVING THEIR ORIGIN AT A PARTICULAR POINT
370                             !   -DIMENSIONS ARE RANGE AND BEAM_BEARING
380   INTEGER Xy_overlay(16380)
390  ! INTEGER Sector_overlay(16380)
400                             !
410   PRINT "MAX_BEARING=";Max_bearing,"BEARING_INC=";Bearing_inc,"BEAM_WIDTH=";
Beam_width,"BEAM_CENTER=";Beam_center
420   PRINT "MIN_RANGE=';Min_range,"MAX_RANGE=";Max_range,"RANGE_INC=";Range_inc
430   PRINT "SOUNDSPEED=";Soundspeed,"BLANK-DISTANCE=";Blank_distance,"PULSE_WID
TH=";Pulse_width
440   PRINT "MAX_PHONE=';Max_phone,"ARRAYLENGTH=";Arraylength,"BASE_INC=";Base_i
nc
450   PRINT LIN(1)
460                             !
470                             !
480                             ! * * * * * * * * * * *
490   PLOTTER IS "GRAPHICS"     ! PLOT FOR IMAGE DISPLAY
500   GRAPHICS
510   SHOW -5.5,5.5,-Max_range-1,-Min_range+1
520  ! GOSUB Sector_grid
530   GCLEAR
540   GOSUB Xy_grid
550                             ! GLOAD Xy_overlay(*) CAN BE USED FROM KEYBOARD
560                             ! GLOAD Sector_overlay(*)  "       "       "
570                             !
580                             ! * * * * * * * * * * *
590                             ! FORM Tracematrix()
600  ! GOSUB Trace
610  ! GOSUB Keep_traces
620   GOSUB Mass_storage        ! TO RETRIEVE PRE-CALCULATED Trace_matrix()
630                             !
```

```
640                             ! * * * * * * * * * * *
650   MAT Objects=(0)           ! SET UP Objects()
660   GOSUB Round_object
670   GOSUB Long_object
680   GOSUB Inclined_object
690                             !
700  ! PRINT LIN(2),"   OBJECTS (Range,Bearing)",LIN(2)
710  ! PRINT USING "31(D,X)/";Objects(*)
720                             !
730                             ! * * * * * * * * * * *
740 Scan:                       ! SCANS A SECTOR
750                             ! PULSE TRANSMITTED AT EACH BEARING
760  FOR Bearing=-Max_bearing TO Max_bearing STEP Bearing_inc
770   B_int=Bearing DIV Bearing_inc !BEARING INDEX FOR Objects() AND Amplitude()
780   B=ABS(B_int)
790                             !
800                             ! * * * * * * * * * * *
810                             ! FORM Indata(Time_slot,Phone)
820   MAT Indata=(0)
830   Obj_element=0             ! BOOLEAN VARIABLE SET IF ELEMENT EXISTS
840                             !
850   R=0
860   FOR Range=Min_range TO Max_range STEP Range_inc
870      IF Objects(R,B_int)=1 THEN GOSUB Receive
880                             ! Receive ALSO SETS Obj_element=1
890      R=R+1
900   NEXT Range
910  ! PRINT LIN(2),"  INDATA (Time_slot,Phone)";LIN(2)
920  ! PRINT USING "16(2D,2X)/";Indata(*)  !CHECK MAX_PHONE
930                             ! END OF FORMING Indata()
940                             !
950                             ! * * * * * * * * * * *
960                             ! FORM Amplitude(Range,Bearing)
970                             !
980   IF Obj_element=0 THEN GOSUB No_target
990   IF Obj_element=1 THEN GOSUB Target_exists
1000 NEXT Bearing
```

```
1010                              !
1020 ! PRINT LIN(2),"  AMPLITUDE (Range,Bearing)",LIN(2)
1030 ! PRINT USING "21(DD,X)/";Amplitude(*)  !CHECK DIM OF Amplitude()
1040                              ! END OF FORMING Amplitude()
1050                              ! * * * * * * * * * * *
1060  BEEP
1070  GOTO Scan                   ! I.E. SCAN CONTINUOUSLY
1080  STOP
1090  END
1100                              ! ----------------------
1110 Trace:                       !              TRACE
1120                              ! CALCULATES THE DISTANCES, PROPAGATION TIMES,
1130                              !   AND THE TRACE FUNCTIONS FOR ELEMENTAL
1140                              !   SCATTERING CENTERS AT RANGES AND BEARINGS
1150                              !   FROM THE CENTER OF A HYDROPHONE LINE-ARRAY
1160                              !
1170                              ! FORM Trace_matrix(B,R,Phone)
1180                              ! LOOP TO FILL Base(Phone) WITH DISTANCES FROM
1190                              !   CENTER OF LINE-ARRAY TO EACH HYDROPHONE
1200  FOR Phone=0 TO Max_phone
1210     IF Phone<=Max_phone/2 THEN Number=Phone
1220     IF Phone>Max_phone/2 THEN Number=Max_phone-Phone
1230     Base(Phone)=Arraylength/2-Number*Base_inc
1240  NEXT Phone
1250                              !
1260                              ! BEARING
1270  B=0                         ! INDEX FOR THE BEARING DIMENSION
1280  FOR Bearing=0 TO Max_bearing STEP Bearing_inc
1290     Cosine_acute=COS(90-Bearing)
1300                              !
1310                              ! RANGE
1320     R=0
1330     FOR Range=Min_range TO Max_range STEP Range_inc
1340 !        PRINT LIN(2),"    BEARING=";Bearing,"RANGE=";Range,LIN(1)
1350 H1:     IMAGE "PHONE",5X,"ARRIVAL_TIME",5X,"TRACE_MATRIX",/
1360 !        PRINT USING H1
```

```
1370 H2:    IMAGE "PHONE",5X,"TRIANGLE_SOLN",5X,"DISTANCE",5X,"ARRIVAL_TIME",5X,
"TRACE_MATRIX",/
1380 !       PRINT USING H2
1390                              !
1400                              ! PHONE
1410        FOR Phone=0 TO Max_phone
1420           IF Phone<=Max_phone/2 THEN Cosine_angle=-Cosine_acute
1430           IF Phone>Max_phone/2 THEN Cosine_angle=Cosine_acute
1440           Triangle_soln=SQR(Range^2+Base(Phone)^2-2*Range*Base(Phone)*Cosin
e_angle)
1450           Distance=Range+Triangle_soln
1460           Arrival_time=(Distance-2*Blank_distance)/Soundspeed
1470           Trace_matrix(B,R,Phone)=Arrival_time DIV Pulse_width
1480 H3:       IMAGE 1X,DD,9X,.DDDDDD,12X,DDD
1490 !          PRINT USING H3;Phone,Arrival_time,Trace_matrix(B,R,Phone)
1500 H4:       IMAGE 1X,DD,10X,DD.DD,12X,DD.DD,8X,.DDDDDD,12X,DDD
1510 !          PRINT USING H4;Phone,Triangle_soln,Distance,Arrival_time,Trace_m
atrix(B,R,Phone)
1520        NEXT Phone
1530        R=R+1
1540     NEXT Range
1550     B=B+1
1560  NEXT Bearing
1570  RETURN
1580                              ! END OF FORMING Trace_matrix()
1590                              !
1600                              ! - - - - - - - - - - - - -
1610 Keep_traces:                 !
1620  CREATE File$,204            ! TO STORE Trace_matrix(15,50,15)
1630                              !   i.e. 2 degree BEARING_INC
1640                              !        0.1 m RANGE_INC
1650  ASSIGN #1 TO File$          !        16 PHONES, FOR 5m LINE ARRAY
1660  PRINT #1;Trace_matrix(*)!     AS A DATA FILE ON DISK
1670  ASSIGN #1 TO *
1680  RETURN
1690                              ! - - - - - - - - - - - - -
```

```
1700 Mass_storage:              ! RETRIEVE PRE-CALCULATED Trace_matrix()
1710  ASSIGN #1 TO File#
1720  READ #1;Trace_matrix(*)
1730  ASSIGN #1 TO *
1740  RETURN
1750                            ! END OF RETRIEVING Trace_matrix()
1760                            !
1770                            ! - - - - - - - - - - - - - -
1780 Receive:                   ! SIMULATE ELEMENT AT RANGE,BEARING
1790 !    PRINT "RANGE=";Range,"BEARING=";Bearing
1800     IF Bearing>=0 THEN GOSUB Bearing_pos
1810     IF Bearing<0 THEN GOSUB Bearing_neg
1820     Obj_element=1
1830     RETURN
1840                            !
1850 Bearing_pos:               !
1860     FOR Phone=0 TO Max_phone
1870        Time_slot=Trace_matrix(B,R,Phone)
1880        Indata(Time_slot,Phone)=Indata(Time_slot,Phone)+1
1890     NEXT Phone
1900     RETURN
1910                            !
1920 Bearing_neg:               !
1930     FOR Phone=0 TO Max_phone
1940        Time_slot=Trace_matrix(B,R,Phone)
1950        Inphone=Max_phone-Phone
1960        Indata(Time_slot,Inphone)=Indata(Time_slot,Inphone)+1
1970     NEXT Phone
1980     RETURN
1990                            ! - - - - - - - - - - - - - -
2000                            ! SETS Amplitude(Range,Bearing)=0 FOR ALL RANGES
2010 No_target:                 !    AT A BEARING
2020     R=0
2030     FOR Range=Min_range TO Max_range STEP Range_inc
2040        IF Amplitude(R,B_int)=1 THEN GOSUB Erasetarget
2050        R=R+1
2060     NEXT Range
2070     RETURN
```

```
2080                            ! - - - - - - - - - - - - - -
2090                            ! FORMS Amplitude() FOR ALL RANGES AT A BEARING
2100 Target_exists:             !
2110     IF Bearing>=0 THEN GOSUB Amplitude_pos
2120     IF Bearing<0 THEN GOSUB Amplitude_neg
2130     RETURN
2140                            !
2150                            ! - - - - - - - - - - - - - -
2160 Amplitude_pos:             ! FORMS Amplitude() ELEMENTS AT RANGES,
2170                            !    POSITIVE BEARING
2180     R=0
2190     FOR Range=Min_range TO Max_range STEP Range_inc
2200        Total=0
2210                            !   ADD THE ELEMENTS FROM Indata()
2220                            !     THAT CORRESPOND TO "RANGE,BEARING"
2230        FOR Phone=0 TO Max_phone
2240           Time_slot=Trace_matrix(B,R,Phone)
2250           Total=Total+Indata(Time_slot,Phone)
2260        NEXT Phone
2270   IF (Total>Threshold) AND (Amplitude(R,B_int)=0) THEN GOSUB Writetarget
2280   IF (Total<=Threshold) AND (Amplitude(R,B_int)=1) THEN GOSUB Erasetarget
2290        R=R+1
2300     NEXT Range
2310     RETURN
2320                            ! - - - - - - - - - - - - - -
2330 Amplitude_neg:             ! FORMS Amplitude() ELEMENTS AT RANGES,
2340                            !    NEGATIVE BEARING
2350     R=0
2360     FOR Range=Min_range TO Max_range STEP Range_inc
2370        Total=0
2380                            !   ADD THE ELEMENTS FROM Indata()
2390                            !     THAT CORRESPOND TO THE MIRROR IMAGE
2400                            !     (wrt Phone) FOR RANGE,BEARING
2410        FOR Phone=0 TO Max_phone
2420           Time_slot=Trace_matrix(B,R,Phone)
2430           Inphone=Max_phone-Phone
2440           Total=Total+Indata(Time_slot,Inphone)
2450        NEXT Phone
```

```
2460   IF (Total>Threshold) AND (Amplitude(R,B_int)=0) THEN GOSUB Writetarget
2470   IF (Total<=Threshold) AND (Amplitude(R,B_int)=1) THEN GOSUB Erasetarget
2480        R=R+1
2490     NEXT Range
2500     RETURN
2510                              !
2520                              ! - - - - - - - - - - - - - -
2530 Writetarget:                 ! WRITE A NEW TARGET
2540  PEN 1
2550  GOSUB Target
2560  Amplitude(R,B_int)=1        ! STORE INFO - CONTACT EXISTS AT (RANGE,BEARING)
2570  RETURN
2580                              !
2590                              ! - - - - - - - - - - - - - - - -
2600 Erasetarget:                 ! ERASE A PREVIOUS TARGET
2610  PEN -1
2620  GOSUB Target
2630  Amplitude(R,B_int)=0        ! STORE INFO - NO CONTACT AT (RANGE,BEARING)
2640  RETURN
2650                              ! - - - - - - - - - - - - - - - -
2660                              ! TARGET SYMBOL
2670 Target:                      !
2680  PDIR Bearing
2690  FOR Y=Range-.04 TO Range+.05 STEP .01
2700     MOVE 0,0
2710     IPLOT -Y*Dx,-Y,-2
2720     IPLOT 2*Y*Dx,0,-1
2730  NEXT Y
2740  RETURN
2750                              ! - - - - - - - - - - - - - - -
2760                              ! DRAW SECTOR GRID
2770 Sector_grid:                 !
2780  UNCLIP
2790  LORG 5
2800  Mark=.1
2810  Line=Mark                   ! DRAW GRID-LINES FOR 10 DEG BEARINGS
2820  FOR Bearing=-Max_bearing TO Max_bearing STEP 10
2830     PDIR Bearing
```

```
2840     GOSUB Marks
2850     IPLOT 0,-.3,-2
2860     LABEL Bearing
2870  NEXT Bearing
2880                              !
2890  Line=Mark/2                 ! DRAW GRID-LINES FOR 5 DEG BEARINGS
2900  FOR Bearing=-Max_bearing+5 TO Max_bearing-5 STEP 10  -
2910     PDIR Bearing
2920     GOSUB Marks
2930  NEXT Bearing
2940                              ! DRAW POINT GRID
2950  Line=Mark/2
2960  FOR Bearing=-Max_bearing TO Max_bearing STEP Max_bearing
2970     PDIR Bearing
2980     FOR Range=Min_range+1 TO Max_range-1
2990        MOVE 0,0
3000        IPLOT 0,-(Range-Line/2),-2
3010        IPLOT 0,-Line,-1
3020     NEXT Range
3030  NEXT Bearing
3040                              ! LABEL RANGES
3050  PDIR -Max_bearing-2
3060  GOSUB Range_label
3070  PDIR Max_bearing+2
3080  GOSUB Range_label
3090                              !
3100                              ! TITLE AXES
3110  PDIR -Max_bearing-5
3120  MOVE 0,0
3130  IPLOT 0,-Min_range-(Max_range-Min_range)/2,-2
3140  LDIR 90-Max_bearing-5
3150  LABEL "RANGE (m)"
3160                              !
3170  PDIR 0
3180  MOVE 0,0
3190  IPLOT 0,-(Max_range+1),-2
3200  LDIR 0
3210  LABEL "BEARING (Deg)"
```

```
3220  GSTORE Sector_overlay(*)
3230  RETURN                    ! END OF DRAWING SECTOR GRID
3240                            !
3250                            !
3260 Marks:                     ! DRAW GRID-LINES
3270  MOVE 0,0
3280  IPLOT 0,-(Min_range-Line),-2
3290  IPLOT 0,-Line,-1
3300  IPLOT 0,-(5-Line),-2
3310  IPLOT 0,-2*Line,-1
3320  RETURN
3330                            ! LABEL RANGES
3340 Range_label:               !
3350  FOR Range=Min_range TO Max_range STEP 1
3360     MOVE 0,0
3370     IPLOT 0,-Range,-2
3380     LABEL Range
3390  NEXT Range
3400  RETURN
3410                            !
3420                            ! - - - - - - - - - - - - - - - -
3430                            ! DRAW XY_GRID
3440 Xy_grid:                   !
3450  CLIP -5,5,-Max_range,-Min_range+1
3460  LINE TYPE 3
3470  GRID 1,1,0,0,1,1
3480  UNCLIP
3490  LINE TYPE 1
3500  LORG 5                    ! LABEL X-AXIS
3510  FOR X=-7 TO 7
3520     MOVE X,-10.5
3530     LABEL X
3540  NEXT X
3550  LORG 8                    ! LABEL Y-AXIS
3560  FOR Y=-Max_range TO -Min_range+1
3570     MOVE -4.9,Y
3580     LABEL Y
3590  NEXT Y
```

```
3600  GSTORE Xy_overlay(*)
3610  RETURN                    ! END OF DRAWING XY_GRID
3620                            ! - - - - - - - - - - - - - -
3630                            ! SIMULATED ROUND TARGET
3640 Round_object:              !
3650  FOR Bearing=-20 TO -16 STEP 2
3660     FOR Range=5.9 TO 6.1 STEP .1
3670        GOSUB Setup_objects
3680     NEXT Range
3690  NEXT Bearing
3700  Bearing=-18
3710      Range=5.8
3720        GOSUB Setup_objects
3730  RETURN
3740                            ! - - - - - - - - - - -
3750                            !
3760                            ! SIMULATED LONG LEVEL TARGET
3770 Long_object:               !
3780  FOR Bearing=-22 TO -20 STEP 2
3790     FOR Range=9.7 TO 9.8 STEP .1
3800        GOSUB Setup_objects
3810     NEXT Range
3820  NEXT Bearing
3830  FOR Bearing=-18 TO -16 STEP 2
3840     FOR Range=9.5 TO 9.6 STEP .1
3850        GOSUB Setup_objects
3860     NEXT Range
3870  NEXT Bearing
3880  FOR Bearing=-14 TO -12 STEP 2
3890     FOR Range=9.3 TO 9.5 STEP .1
3900        GOSUB Setup_objects
3910     NEXT Range
3920  NEXT Bearing
3930  FOR Bearing=-10 TO -8 STEP 2
3940     FOR Range=9.2 TO 9.4 STEP .1
3950        GOSUB Setup_objects
3960     NEXT Range
3970  NEXT Bearing
```

```
3980  FOR Bearing=-6 TO -4 STEP 2
3990     FOR Range=9.1 TO 9.3 STEP .1
4000        GOSUB Setup_objects
4010     NEXT Range
4020  NEXT Bearing
4030  Bearing=-20
4040    Range=9.6
4050       GOSUB Setup_objects
4060  Bearing=-22
4070    Range=9.9
4080       GOSUB Setup_objects
4090  Bearing=-18
4100    Range=9.7
4110       GOSUB Setup_objects
4120  Bearing=-16
4130    Range=9.4
4140       GOSUB Setup_objects
4150  RETURN                    ! END OF LONG LEVEL TARGET
4160                            !
4170                            ! - - - - - - - - - - -
4180                            ! SIMULATED INCLINED TARGET
4190 Inclined_object:           !
4200  Bearing=8
4210    FOR Range=7.4 TO 7.6 STEP .1
4220      GOSUB Setup_objects
4230    NEXT Range
4240  Bearing=10
4250    FOR Range=7.3 TO 7.5 STEP .1
4260      GOSUB Setup_objects
4270    NEXT Range
4280  Bearing=12
4290    FOR Range=7.2 TO 7.4 STEP .1
4300      GOSUB Setup_objects
4310    NEXT Range
4320  Bearing=14
4330    FOR Range=7.1 TO 7.3 STEP .1
4340      GOSUB Setup_objects
4350    NEXT Range
```

```
4360  Bearing=16
4370    FOR Range=7 TO 7.2 STEP .1
4380      GOSUB Setup_objects
4390    NEXT Range
4400  Bearing=18
4410    FOR Range=6.9 TO 7.1 STEP .1
4420      GOSUB Setup_objects
4430    NEXT Range
4440  Bearing=20
4450    FOR Range=6.8 TO 7 STEP .1
4460      GOSUB Setup_objects
4470    NEXT Range
4480  RETURN                    ! END OF SIMULATED INCLINED TARGET
4490                            !
4500                            ! - - - - - - - - - - -
4510                            ! SIMULATE TARGET ELEMENT AT RANGE,BEARING
4520                            !    IN Objects()
4530 Setup_objects:             !
4540  R=(Range-Min_range) DIV Range_inc  !INDEX FOR RANGE DIMENSION
4550  B=Bearing DIV Bearing_inc   !INDEX FOR BEARING DIMENSION
4560  Objects(R,B)=1
4570  RETURN
4580                            ! - - - - - - - - - - -
4590                            ! * * * * * * * * * * *
```

```
 1 /*  PLI-80 VERSION -- IMAGES.PLI          30 OCT 80  */
 2 images: proc options(main);
 3      dcl (A_indata(0:2671), B_indata(0:2671), A_brg, B_brg) fixed(7),
 4          (b_array(0:15), r_array(0:50), pn_array(0:15),
 5          trace(0:13055), amp_array(-15:15), amplitude(0:1580)) fixed(15),
 6          (A_data, B_data, A_ampl, B_ampl) bit(1),
 7          (i, b_index) fixed(7),
 8          traces file;
 9
10      %replace
11         arraylength by 5E0,          /*  LENGTH OF HYDROPHONE ARRAY  (m) */
12         max_phone_index by 15,       /*  CORRESPONDS TO 16 PHONES */
13         min_range by 5.0E0,          /*  ALL RANGES IN METERS */
14         max_range by 10.0E0,
15         range_inc by 0.1E0,
16         max_range_index by 50,       /*  51 RANGE ELEMENTS */
17         soundspeed by 1500E0,        /*  METERS/SEC  */
18         pulse_width by 50E-6,        /*  50 MICROSECONDS  */
19         blank_distance by 4.5E0,
20         max_bearing by 30,           /*  DEGREES  */
21         bearing_inc by 2,
22         max_brg_index by 15,         /*  16 BEARING ELEMENTS */
23         max_timeslot by 166,         /*  167 DATA SAMPLES */
24         read_disk by '1'b;           /*  BOOLEAN VARIABLE CONTROLS CALCULATING
25                                       TRACE() OR READING IT FROM A DISK FILE */
26
27                          /* INITIALIZE SEMAPHORES */
28         A_ampl = '0'b; B_ampl = '0'b;    /* UNLOCKED */
29         A_data = '1'b; B_data = '1'b;    /* LOCKED   */
30
```

```
31          do;
32            call setup();
33            put skip list ('SETUP COMPLETED');
34
35            do i = 0 to 10;                    /* SCAN A NUMBER OF TIMES */
36               do b_index = -max_brg_index to max_brg_index by 2;
37                  call indata();
38                  call sum_amplitudes();
39               end;                            /*  ONE COMPLETE SCAN  */
40            end;                        /*  A NUMBER OF SCANS  */
41
42            call see_amplitude(-15,0);
43            call see_amplitude(0,15);
44          end;
45
46                  /* ================================== */
47
48    setup: proc;
49          dcl (base(0:15), base_inc, range, max_phone) float,
50              (cosine_angle, cosine_acute) float,
51              (triangle_soln, distance, arrival_time) float,
52              (i, phone, b_index, r_index, bearing) fixed(7),
53              (number, timeslot, brg_address, b_r_addr) fixed(15);
54
55                  /*  FILL B_ARRAY WITH ADDRESSES CORRESPONDING
56                     TO THE BEARING DIMENSION OF TRACE_MATRIX */
57          number = (max_phone_index + 1)*(max_range_index + 1);
58          do i = 0 to max_brg_index;
59             b_array(i) = i*number;
60          end;
61
62                  /*  FILL R_ARRAY WITH ADDRESSES CORRESPONDING
63                     TO THE RANGE DIMENSION OF TRACE_MATRIX */
64          do i = 0 to max_range_index;
65             r_array(i) = i*(max_phone_index + 1);
66          end;
```

```
67
68                /*  FILL PH_ARRAY WITH ADDRESSES CORRESPONDING
69                    TO THE PHONE DIMENSION OF INDATA( PHONE, TIME ) */
70          do i = 0 to max_phone_index;
71             ph_array(i) = i*(max_timeslot + 1);
72          end;
73
74                /*  FILL AMP_ARRAY WITH ADDRESSES CORRESPONDING
75                  TO THE BEARING DIMENSION OF THE AMPLITUDE MATRIX */
76          i = 0;
77          do b_index = -max_brg_index to max_brg_index;
78             amp_array(b_index) = i*(max_range_index + 1);
79             i = i + 1;
80          end;
81
82                /*  GET DATA FOR TRACE()  */
83          if read_disk then call read_traces();
84          else call fill_traces();
85
86      /*  do bearing = 0 to 30 by 10;
87             do range = 5.0 to 10.0;
88               call see_trace(bearing,range);
89             end;
90          end;    */
91
92                /*  SIMULATE CONTENTS OF THE INDATA MATRICES
93                    AS THOUGH ORIGINATED BY SCATTERERS AT
94                    PARTICULAR RANGES AND BEARINGS   */
95          do;
96          call receive(A_indata, -30, 5.0);
97          call receive(A_indata, 10, 9.0);
98          call receive(B_indata, -20, 6.0);
99          call receive(B_indata, 20, 10.0);
100         end;
101
```

```
102     /*   do;
103          put skip(2) list('A_indata');
104          call see_data(A_indata);
105          put skip(2) list('B_indata');
106          call see_data(B_indata);
107          end;    */
108
109                  /* ------------------------ */
110
111                  /*  FILL TRACE() WITH THE TIMESLOT INDICES THAT
112                    CORRESPOND TO THE ARRIVAL TIME AT EACH HYDROPHONE
113                    OF THE CURVED WAVEFRONTS FROM ELEMENTAL SCATTERERS
114                    AT EACH OF THE RANGES AND BEARINGS OF INTEREST  */
115
116          fill_traces: proc;
117                  /*  FILL BASE(PHONE) WITH THE DISTANCES FROM
118                    THE CENTER OF THE LINE ARRAY TO EACH HYDROPHONE */
119            max_phone = max_phone_index;
120            base_inc = arraylength/max_phone;
121            do phone = 0 to max_phone_index;
122              if phone > max_phone/2 then
123                  number = max_phone_index - phone;
124              else number = phone;
125              base(phone) = arraylength/2 - number*base_inc;
126            end;
127
128            do b_index = 0 to max_brg_index;
129               bearing = b_index*bearing_inc;
130               brg_address = b_array(b_index);
131                  /* put skip list ('bearing =',bearing); */
132               cosine_acute = cosd(90-bearing);
133
134               do r_index = 0 to max_range_index;
135                  range = r_index*range_inc + min_range;
136                  b_r_addr = r_array(r_index) + brg_address;
```

```
137
138                do phone = 0 to max_phone_index;
139                   if phone > max_phone/2 then
140                        cosine_angle = cosine_acute;
141                   else   cosine_angle = -cosine_acute;
142                   triangle_soln = sqrt(range*range+base(phone)*base(phone)
143                                  -2*range*base(phone)*cosine_angle);
144                   distance = range + triangle_soln;
145                   arrival_time = (distance - 2*blank_distance)/soundspeed;
146                   trace(b_r_addr + phone) = arrival_time/pulse_width;
147                end;
148             end;
149          end;
150          open file (traces) output sequential keyed
151                env(f(26112),b(128));
152          write file (traces) from (trace);
153          close file (traces);
154       end fill_traces;
155
156                /*  FILL TRACE() FROM A PRE-CALCULATED FILE
157                   TRACES.DAT STORED ON DISK  */
158       read_traces: proc;
159          open file (traces) input sequential keyed
160                env(f(26112),b(128));
161          read file (traces) into (trace);
162          close file (traces);
163       end read_traces;
164
165       see_trace: proc(brg,range);
166                dcl (brg , range) float;
167
168                b_index = brg/bearing_inc;
169                r_index = (range - min_range)/range_inc;
170                b_r_addr = b_array(b_index) + r_array(r_index);
171                put skip(2) list ('brg=',brg,'range=',range,
172                                  'b_r_addr=',b_r_addr);
```

```
173                 put skip edit ((trace(b_r_addr + phone)
174                         do phone=0 to max_pnone_index))
175                         (16(f(3),x(1)));
176         end see_trace;
177
178         receive: proc(data_array, brg, range);
179                 dcl data_array(0:2671) fixed(7),
180                     (brg, range) float;
181
182            b_index = abs(brg/bearing_inc);
183            r_index = (range - min_range)/range_inc;
184            b_r_addr = b_array(b_index) + r_array(r_index);
185            if brg < 0 then call brg_neg();
186            else call brg_pos();
187
188            brg_pos: proc;
189                 do phone = 0 to max_phone_index;
190                    timeslot = trace(b_r_addr + phone);
191                    data_array(ph_array(phone) + timeslot) = 1;
192                 end;
193            end brg_pos;
194
195            brg_neg: proc;
196                 do phone = 0 to max_phone_index;
197                    timeslot = trace(b_r_addr + phone);
198                    i = max_phone_index - phone;
199                    data_array(ph_array(i) + timeslot) = 1;
200                 end;
201            end brg_neg;
202
203         end receive;
204
```

```
205         see_data: proc(data_array);
206                 dcl data_array(0:2671) fixed(7);
207
208            put skip;
209            do timeslot = 0 to max_timeslot;
210                 put skip edit((data_array(pn_array(phone)+timeslot)
211                         do phone = 0 to max_phone_index))
212                         (16(f(3),x(1)));
213            end;
214         end see_data;
215
216     end setup;
217
218                 /* ============================== */
219
220                 /*  CREATES THE A_INDATA AND B_INDATA ARRAYS WHICH
221                   CONTAIN THE TIME SAMPLES OF THE SIGNAL AMPLITUDES
222                   AT EACH OF THE HYDROPHONES AFTER EACH TRANSMITTED
223                   ACOUSTIC PULSE  */
224     indata: proc;
225
226        do;
227          call wait(A_amp1);
228          A_data = '1'b;          /* BUSY WITH THE A_INDATA ARRAY */
229          A_brg = b_index;
230          call transmit_and_sample(A_indata);
231          A_data = '0'b;          /* PROCESS FINISHED WITH A_INDATA */
232
233          call wait(B_amp1);
234          B_data = '1'b;          /* BUSY WITH THE B_INDATA ARRAY */
235          B_brg = b_index + 1;
236          call transmit_and_sample(B_indata);
237          B_data = '0'b;          /* PROCESS FINISHED WITH B_INDATA */
238        end;
239
240
```

```
241                 /* ---------------------- */
242
243       wait: proc(busy);
244         dcl busy bit(1);
245         do while (busy);
246         end;
247         busy = '1'b;              /* RESET SEMAPHORE */
248       end wait;
249
250       transmit_and_sample: proc(data_array);
251         dcl data_array(0:2671) fixed(7);
252
253         do;
254         end;
255
256       end transmit_and_sample;
257
258   end indata;
259
260                 /* ============================== */
261
262                 /*  CREATES THE AMPLITUDE ARRAY - EACH ELEMENT OF WHICH
263                   REPRESENTS THE TOTAL SIGNAL AMPLITUDE FROM SCATTERERS
264                   AT EACH OF THE RANGES AND BEARINGS OF INTEREST  */
265   sum_amplitudes: proc;
266
267       do;
268         call test(A_data);        /* WAIT FOR A_INDATA TO BE READY */
269         call form_amplitude(A_brg, A_indata);
270         A_ampl = '0'b;            /* PROCESS FINISHED WITH A_INDATA */
271
272         call test(B_data);        /* WAIT FOR B_INDATA TO BE READY */
273         call form_amplitude(B_brg, B_indata);
274         B_ampl = '0'b;            /* PROCESS FINISHED WITH B_INDATA */
275       end;
276
```

```
277                 /* -------------------- */
278
279         test: proc(busy);
280           dcl busy bit(1);
281           do while (busy);
282           end;
283         end test;
284
285                 /*  SUMS THE ELEMENTS OF THE INDATA ARRAYS THAT ARE
286                   IDENTIFIED BY TRACE() AS CORRESPONDING TO THE
287                   WAVEFRONT CURVATURE AND ARRIVAL TIMESLOT FOR EACH
288                   OF THE RANGES AT A PARTICULAR BEARING  */
289         form_amplitude: proc(b_index, data_array);
290           dcl (b_index, data_array(0:2671),
291                r_index, phone) fixed(7),
292               (brg_address, b_r_addr, amp_addr, total) fixed(15);
293
294           do;
295             brg_address = b_array( abs(b_index) );
296             amp_addr = amp_array( b_index );
297             if b_index < 0 then call amp_neg();
298             else call amp_pos();
299           end;
300
301           amp_pos: proc;
302             do r_index = 0 to max_range_index;
303               b_r_addr = r_array( r_index ) + brg_address;
304               total = 0;
305               do phone = 0 to max_phone_index;
306                 total = total + data_array( ph_array(phone) +
307                             trace( b_r_addr + phone ));
308               end;
309               amplitude( amp_addr + r_index ) = total;
310             end;
311           end amp_pos;
312
```

```
313         amp_neg: proc;
314           do r_index = 0 to max_range_index;
315             b_r_addr = r_array( r_index ) + brg_address;
316             total = 0;
317             do phone = 0 to max_phone_index;
318               total = total + data_array( ph_array(max_phone_index-phone)+
319                           trace( b_r_addr + phone ));
320             end;
321             amplitude( amp_addr + r_index ) = total;
322           end;
323         end amp_neg;
324
325       end form_amplitude;
326
327    end sum_amplitudes;
328
329                 /* ================================ */
330
331     see_amplitude: proc(start,finish);
332         dcl (start, finish, b_index, r_index) fixed(7);
333
334         put skip(2) list ('AMPLITUDE FROM',start,'   TO',finish,
335                                    BRG_INDEX');
336         put skip edit((b_index   do b_index=start to finisn))
337                 (16(f(3),x(1)));
338         put skip;
339         do r_index = 0 to max_range_index;
340            put skip edit((amplitude(amp_array(b_index) + r_index)
341                 do b_index = start to finish))  (16(f(3),x(1)));
342         end;
343     end see_amplitude;
344
345                 /* ================================ */
346
347 end images;
```

```
/*  OPERATIONAL VERSION -- IMTAC.PLI.....7 NOV 80  */
images: proc options(main);
     dcl dummy_array(0:767) fixed(7);
     dcl (A_indata(0:2671), B_indata(0:2671), A_brg, B_brg) fixed(7),
         (b_array(0:15), r_array(0:50), pn_array(0:15),
         trace(0:13655), amp_array(-15:15), amplitude(0:1580)) fixed(15),
         (A_data, B_data, A_amp1, B_amp1, A_amp2, B_amp2,
          A_amp3, B_amp3) bit(1);

     %replace
        arraylength by 5E0,          /*  LENGTH OF HYDROPHONE ARRAY  (m) */
        max_phone_index by 15,       /*  CORRESPONDS TO 16 PHONES */
        min_range by 5.0E0,          /*  ALL RANGES IN METERS */
        max_range by 10.0E0,
        range_inc by 0.1E0,
        max_range_index by 50,       /*  51 RANGE ELEMENTS */
        soundspeed by 1500E0,        /*  METERS/SEC  */
        pulse_width by 50E-6,        /*  50 MICROSECONDS  */
        blank_distance by 4.5E0,
        max_bearing by 30,           /*  DEGREES  */
        bearing_inc by 2,
        max_brg_index by 15,         /*  16 BEARING ELEMENTS */
        max_timeslot by 166;         /*  167 DATA SAMPLES */

                              /* INITIALIZE SEMAPHORES */
     A_amp1 = '0'b; B_amp1 = '0'b;    /* UNLOCKED */
        A_amp2 = '0'b; B_amp2 = '0'b;
        A_amp3 = '0'b; B_amp3 = '0'b;
        A_data = '1'b; B_data = '1'b;    /* LOCKED   */
```

```
30
31          do;
32            call setup();
33            call indata();
34            call sum_amplitudes();
35          end;
36
37                  /* ================================== */
38
39      setup: proc;
40          dcl (i, b_index) fixed(7),
41              number fixed(15);
42
43                  /*  FILL B_ARRAY WITH ADDRESSES CORRESPONDING
44                    TO THE BEARING DIMENSION OF TRACE_MATRIX */
45          number = (max_phone_index + 1)*(max_range_index + 1);
46          do i = 0 to max_brg_index;
47             b_array(i) = i*number;
48          end;
49
50                  /*  FILL R_ARRAY WITH ADDRESSES CORRESPONDING
51                     TO THE RANGE DIMENSION OF TRACE_MATRIX */
52          do i = 0 to max_range_index;
53             r_array(i) = i*(max_phone_index + 1);
54          end;
55
56                  /*  FILL PH_ARRAY WITH ADDRESSES CORRESPONDING
57                     TO THE PHONE DIMENSION OF INDATA( PHONE, TIME ) */
58          do i = 0 to max_phone_index;
59             ph_array(i) = i*(max_timeslot + 1);
60          end;
61
```

```
62                /*  FILL AMP_ARRAY WITH ADDRESSES CORRESPONDING
63                  TO THE BEARING DIMENSION OF THE AMPLITUDE MATRIX */
64          i = 0;
65          do b_index = -max_brg_index to max_brg_index;
66             amp_array(b_index) = i*(max_range_index + 1);
67             i = i + 1;
68          end;
69
70    end setup;
71
72                /* ============================== */
73
74                /*  CREATES THE A_INDATA AND B_INDATA ARRAYS WHICH
75                  CONTAIN THE TIME SAMPLES OF THE SIGNAL AMPLITUDES
76                  AT EACH OF THE HYDROPHONES AFTER EACH TRANSMITTED
77                  ACOUSTIC PULSE  */
78    indata: proc;
79      dcl (i, b_index) fixed(7);
80
81      do i = 0 to 10;              /* SCAN A NUMBER OF TIMES */
82        do b_index = -max_brg_index to (max_brg_index - 1) by 2;
83
84          call wait(A_amp1);
85          call wait(A_amp2);
86          call wait(A_amp3);
87          A_data = '1'b;           /* BUSY WITH THE A_INDATA ARRAY */
88          A_brg = b_index;
89          call transmit_and_sample(A_indata);
90          A_data = '0'b;           /* PROCESS FINISHED WITH A_INDATA */
91
92          call wait(B_amp1);
93          call wait(B_amp2);
94          call wait(B_amp3);
95          B_data = '1'b;           /* BUSY WITH THE B_INDATA ARRAY */
96          B_brg = b_index + 1;
97          call transmit_and_sample(B_indata);
```

```
 98          B_data = '0'b;              /* PROCESS FINISHED WITH B_INDATA */
 99
100        end;                         /*  ONE COMPLETE SCAN  */
101      end;                           /*  A NUMBER OF SCANS  */
102
103
104                /* ---------------------- */
105
106       wait: proc(busy);
107         dcl busy bit(1);
108         do while (busy);
109         end;
110         busy = '1'b;                /* RESET SEMAPHORE */
111       end wait;
112
113       transmit_and_sample: proc(data_array);
114         dcl data_array(0:2671) fixed(7);
115
116         do;
117         end;
118
119       end transmit_and_sample;
120
121    end indata;
122
123                /* ============================== */
124
125                /*  CREATES THE AMPLITUDE ARRAY - EACH ELEMENT OF WHICH
126                  REPRESENTS THE TOTAL SIGNAL AMPLITUDE FROM SCATTERERS
127                  AT EACH OF THE RANGES AND BEARINGS OF INTEREST  */
128    sum_amplitudes: proc;
129
130       do while ('1'b);              /* LOOP FOREVER */
131         call test(A_data);          /* WAIT FOR A_INDATA TO BE READY */
132         call form_amplitude(A_brg, A_indata);
133         A_ampl = '0'b;              /* PROCESS FINISHED WITH A_INDATA */
```

```
134
135         call test(B_data);        /* WAIT FOR B_INDATA TO BE READY */
136         call form_amplitude(B_brg, B_indata);
137         B_ampl = '0'b;            /* PROCESS FINISHED WITH B_INDATA */
138       end;
139
140               /* ---------------------- */
141
142       test: proc(busy);
143         dcl busy bit(1);
144         do while (busy);
145         end;
146       end test;
147
148               /*  SUMS THE ELEMENTS OF THE INDATA ARRAYS THAT ARE
149                   IDENTIFIED BY TRACE() AS CORRESPONDING TO THE
150                   WAVEFRONT CURVATURE AND ARRIVAL TIMESLOT FOR EACH
151                   OF THE RANGES AT A PARTICULAR BEARING  */
152       form_amplitude: proc(b_index, data_array);
153         dcl (b_index, data_array(0:2671),
154              r_index, phone) fixed(7),
155             (brg_address, b_r_addr, amp_addr, total) fixed(15);
156
157         do;
158           brg_address = b_array( abs(b_index) );
159           amp_addr = amp_array( b_index );
160           if b_index < 0 then call amp_neg();
161           else call amp_pos();
162         end;
163
164         amp_pos: proc;
165           do r_index = 0 to 50;
166             b_r_addr = r_array( r_index ) + brg_address;
167             total = 0;
```

```
168             do phone = 0 to max_phone_index;
169               total = total + data_array( pn_array(phone) +
170                         trace( b_r_addr + phone ));
171             end;
172             amplitude( amp_addr + r_index ) = total;
173           end;
174         end amp_pos;
175
176         amp_neg: proc;
177           do r_index = 0 to 50;
178             b_r_addr = r_array( r_index ) + brf_address;
179             total = 0;
180             do phone = 0 to max_phone_index;
181               total = total + data_array( pn_array(max_phone_index-phone)+
182                         trace( b_r_addr + phone ));
183             end;
184             amplitude( amp_addr + r_index ) = total;
185           end;
186         end amp_neg;
187
188       end form_amplitude;
189
190     end sum_amplitudes;
191
192                 /* ============================== */
193
194 end images;
```

LIST OF REFERENCES

1. Sutton, J.L., "Underwater Acoustic Imaging", Proceedings of the IEEE, Vol. 67, No. 4, p. 554-566, April 1979.

2. Moffett, M.B., Westervelt, P.J., and Beyer, R.T., "Large-Amplitude Pulse Propagation - A Transient Effect", J. Acoust. Soc. Am., Vol. 49, No. 1, p. 339-343, January 1971.

3. Sackman, G.L., and Shelef, S.C., "The Use of Time-Delay/Phase Difference Trace Functions for Bearing Estimation in Arrays", Proc. Time Delay Estimation and Applications Conference, Vol. 1, (J.C. Hassab, Ed.) Naval Underwater Systems Center, Newport, R.I., July 1979.

4. Hewlett-Packard System 45B Desktop Computer, Operating and Programming Manual (09845B-91000), 1979, (Hewlett-Packard Desktop Computer Division, 3404 East Harmony Road, Fort Collins, Colorado 80525).

5. SBC 80/20 and SBC 80/20-4 Single Board Computers Hardware Reference Manual, Publication No. 98-317C, 1977, (Intel Corporation, Customer Services, 3065 Bowers Ave., Santa Clara, CA, 95051).

6. PL/I-80 Reference Manual, 1980, (Digital Research, P.O. Box 579, Pacific Grove, CA, 93950).

7. Intellec 800 Microcomputer Development System (MDS) Operators Manual, Publication No. 98-129A, 1975, (Intel Corporation, Customer Services, 3065 Bowers Ave., Santa Clara, CA, 95051).

8. An Introduction to CP/M Features and Facilities, 1978, (Digital Research, P.O. Box 579, Pacific Grove, CA, 93950).

9. CP/M Dynamic Debugging Tool (DDT) Users Guide, 1978, (Digital Research, P.O. Box 579, Pacific Grove, CA, 93950).

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 61 Department of Physics and Chemistry Naval Postgraduate School Monterey, California 93940	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	Associate Professor G.L. Sackman, Code 62Sa Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	15
6.	Associate Professor U.R. Kodres, Code 52Kr Department of Computer Science Naval Postgraduate School Monterey, California 93940	2
7.	DACS National Defence Headquarters Ottawa, Ontario Canada K1A 0K2	1
8.	LCol J.C. Bauer, DACS 3 National Defence Headquarters Ottawa, Ontario Canada K1A 0K2	2
9.	Capt G.R. Vermander 2130 Blue Jay Cres. Ottawa, Ontario Canada K1J 6B1	1

Thesis
V415
c.1

190773

Vermander
A signal processing algorithm based on multiple microprocessors for an underwater acoustic imaging system.